



HP Articles Forum

[[New Article](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
 [[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) |
[Ads without Auctions](#)]

Welcome back!
 Your last visit began on 30 Oct 2014, 12:55 p.m.
 Since then, no new articles have been posted.

This forum is intended to allow users to contribute more comprehensive articles than those that are typically found in the general forums. This could include information such as advanced programming and repair instructions as well thoughts about specific calculators or HP calculators in general. Please choose your titles carefully so your information can be easily found.

Please do **not** ask questions here - it is not possible to reply to the messages in this forum. If you want to refer to one of these articles in the general forum, please refer to it by name or article number Please use the [HP Calculator Ads](#) section for "for sale" and "wanted" ads. Please review the [Terms of Use](#) for more complete information.

It's possible to use some [advanced formatting techniques](#) in this forum if you wish.

Article Index

Articles Posted or Modified Within the Last Archive

*210 of 210 Articles Displayed
 (Reversed Threaded Listing)*

- [1266: HP-41 MCODE: The Last Function - at last!](#), Ángel Martin - 8 Nov 2013, 4:55 a.m.
- [1263: HP-80 History, Design and Interesting Facts](#), Max Stone - 24 Oct 2013, 4:14 p.m.
- [1253: RPN-1200](#), Benoit Maag - 25 Aug 2013, 1:02 a.m.
- [1252: 41 SOLVE & INTEG Bank-switched Implementation.](#), Ángel Martin - 21 June 2013, 11:12 a.m.
- [1246: Computing Sunrise/Sunset with the HP 35s](#), Justin Sloan (Honolulu) - 29 Apr 2013, 12:30 a.m.
- [1240: HP-42S with Electroluminescent screen and simple I/O port](#), Jose Poyan - 26 Mar 2013, 4:45 p.m.
- [1220: Sudoku Solver for the HP-15c](#), Marcel Samek - 4 Feb 2013, 1:20 p.m.
- [1218: Mapping HP48 Text to Unicode](#), Chris Dreher - 16 Jan 2013, 1:15 a.m.
- [1205: HP-41 "Halfnut": patching the ROM for full lower case character set display](#), J-F Garnier - 28 Dec 2012, 12:25 p.m.
- [1200: 41-CL Routine of the week: MMUSWAP](#), Ángel Martin - 4 Nov 2012, 5:57 a.m.
- [1197: Palmer's Matrix Multi-Tool Extension](#), mbrethen - 30 Sept 2012, 4:43 p.m.
- [1196: HP-UX and RPL/2](#), BERTRAND Joël - 26 Sept 2012, 6:03 a.m.
- [1195: 41-MCODE: Breaking the FAT Barrier](#), Ángel Martin - 3 Sept 2012, 6:16 a.m.
- [1183: Make PIL-Box ILPer program run on OS X](#), Juergen Keller - 2 Sept 2012, 4:24 p.m.
- [1179: HP-82163A how-to and video](#), Sylvain Cote - 21 July 2012, 2:47 p.m.
- [1174: HP-41 Post Replacement](#), aj04062 - 12 July 2012, 9:45 p.m.
- [1173: 41-MCODE: Dr. Jekyll & Mr. Hyde](#), Ángel Martin - 7 July 2012, 3:48 a.m.
- [1172: Calculation of factorials using logarithms \(HP41 MCODE\)](#), Frido Bohn - 26 May 2012, 9:15 a.m.
- [1162: 41CL - port configuration](#), Christoph Klug - 22 Mar 2012, 5:14 p.m.
- [1158: HP-41 MCODE: Checking ROMs Configuration](#), Ángel Martin - 18 Mar 2012, 3:23 a.m.

- 1152: HP 17BII+ TVM Solver Solution expressing the Balloon Payment as a Percentage with Begin and End mode [and Down Payment as a percentage]**, bmaskell - 3 Mar 2012, 12:36 a.m.
- 1150: Resurrecting a shattered HP 19B**, Neil Hamilton (Ottawa) - 30 Jan 2012, 2:14 p.m.
- 1149: HP41 Trip Computer**, Christopher Johnson - 29 Jan 2012, 9:36 p.m.
- 1133: [WP34s] The ultimate Polynomial RootSolver for the WP34s**, fhub - 3 Jan 2012, 7:23 a.m.
- 1131: RPN Sunrise / Sunset programs for HP41C/HP42S/HP33S/HP35S**, M. Joury - 14 Dec 2011, 9:25 a.m.
- 1129: Incomplete Gamma Function for wp34s**, Les Wright - 5 Dec 2011, 2:40 a.m.
- 1126: Romberg Integration on the wp34s**, Les Wright - 4 Dec 2011, 7:43 p.m.
- 1118: WP 34S: Polynomial root finder adapted from Valentin Albillo's HP 41 program**, Miguel Toro - 23 Nov 2011, 7:32 p.m.
- 1117: WP 34S: financial routines IRR, NPV, Amortization, Interest rate conversion (updated 2013-04-28)**, Miguel Toro - 12 Nov 2011, 2:20 p.m.
- 1113: Inside Saturn Emulator Timing**, Christoph Giesselink - 20 Oct 2011, 4:56 p.m.
- 1107: HP 15 LE: Errata list for Owner's Handbook**, Felix Gross - 10 Oct 2011, 3:03 a.m.
- 1104: WP 34S: Integration wrapper**, Miguel Toro - 9 Oct 2011, 10:26 p.m.
- 1102: WP 34S: MIO matrix helper * edited***, Miguel Toro - 8 Oct 2011, 8:53 p.m.
- 1096: HP35s date functions: DOW, DDate, Date+, Gregorian to Julian conversions**, David Persinger (US) - 16 Sept 2011, 2:21 p.m.
- 1089: HP-15C LE (Limted Edition) Bug Reports**, Katie Wasserman - 16 Sept 2011, 11:22 a.m.
- 1088: Simple TVM Routines for HP-15C**, Chris McCormack - 16 Sept 2011, 8:46 a.m.
- 1072: Creating ROM image files of an HP-41 module**, Juergen Keller - 9 Aug 2011, 4:55 p.m.
- 1062: Thank You HP Users**, LouisD - 9 Aug 2011, 3:21 p.m.
- 1048: EE calculations, some not obvious (reformatted)**, Thomas Klemm - 26 May 2011, 5:14 p.m.
- 1038: Multiple Period Percentage Change Program for the 30b**, Mark Harman - 25 May 2011, 12:13 a.m.
- 1037: HP 30b programs for interest rate conversions with negative or fractional periods per year**, gene wright - 3 May 2011, 2:30 p.m.
- 1035: Game for HP-35s**, Paul Dale - 16 Mar 2011, 5:46 a.m.
- 1025: N-queens benchmark and the HP-17BII**, Thomas Klemm - 7 Feb 2011, 5:48 a.m.
- 1024: Sum of Digits Divisibility Challenge - an unusual solution**, David Hayden - 16 Jan 2011, 9:39 p.m.
- 1021: How to use a formula in a post**, Thomas Klemm - 8 Nov 2010, 4:11 p.m.
- 1014: Sunrise and Sunset**, Thomas Klemm - 28 Oct 2010, 12:05 p.m.
- 1003: HP 35s. Quadratic Equation Solver (with complex solutions support)**, Pablo P (Spain) - 2 July 2010, 3:53 p.m.
- 1002: Calculator Benchmark: Loops of addition**, gene wright - 2 June 2010, 2:51 p.m.
- 1001: The Caves - Reloaded**, Frido Bohn - 6 May 2010, 5:03 a.m.
- 991: Table of HP48 serial numbers and ROM versions**, megarat - 13 Apr 2010, 12:22 p.m.
- 990: HP 82490A Translator Pac Overlay**, Allen - 8 Feb 2010, 11:32 a.m.
- 989: R->P and P->R for the 35s, Reloaded**, Frido Bohn - 26 Jan 2010, 4:06 a.m.
- 986: Psychrometric calculations on the HP 35s (and 50g)**, Maarten Ambaum (Reading, UK) - 22 Jan 2010, 11:14 a.m.
- 983: R->P & P->R for the 35s**, Paul Dale - 21 Jan 2010, 5:22 a.m.
- 982: nCr and nPr - a 41-MCODE implementation**, Ángel Martin - 27 Nov 2009, 10:46 a.m.
- 980: Fast DiGamma Function for Positive and Negative Real Arguments**, Steven Thomas Smith - 26 Sept 2009, 6:06 a.m.
- 978: Fast Incomplete Gamma/Exponential Integral/Error Functions With Lentz's Method**, Steven Thomas Smith - 23 Sept 2009, 1:28 p.m.
- 944: EIGEN: Characteristic equation+eigenvalues for order 2,3,4,5**, Steven Thomas Smith - 1 Sept 2009, 12:39 p.m.
- 941: GAMMA and GAMMLN: Fast, Accurate Lanczos Approximation for Gamma Functions**, Steven Thomas Smith - 1 Sept 2009, 12:49 p.m.
- 937: Enhance your HP-97 -- Add a Clock and More**, Katie Wasserman - 1 Sept 2009, 6:53 p.m.
- 928: HP41 Prime Factorization Revisited**, Ángel Martin - 27 July 2009, 3:58 a.m.
- 926: Mental juggling: Calculating the day of week**, Thomas Klemm - 7 July 2009, 1:36 p.m.
- 925: A non destructive (easy) method for opening a spice calculator.**, Geoff Quickfall - 27 May 2009, 1:47

p.m.

923: HP 41 Battery posts, another repair, Geoff Quickfall - 20 Apr 2009, 2:58 p.m.

910: Power for HP82162A Printer and HP9114 Disc Drive, Philip Isaac (Canada) - 3 Mar 2009, 12:02 p.m.

909: INIT-Module for ERAMCO RSU-2 Rambox, PeterP - 18 Feb 2009, 6:37 p.m.

908: HP-71B Math RAM, Marcus von Cube, Germany - 16 Feb 2009, 8:06 a.m.

903: HP41 - How to mix and match UCODE and MCODE in a module, PeterP - 13 Feb 2009, 7:37 p.m.

902: cheaper batteries for the HP41CX, Michael Harwood - 30 Jan 2009, 9:32 p.m.

901: HP9815 - repairing the paper advance mechanism, Mark Edmonds - 19 Oct 2008, 8:29 a.m.

899: Many Digits of Pi (HP32sii, HP-16c, HP-12c, HP-12cp, HP-19c, HP-30b, HP-15c), Katie Wasserman - 17 Oct 2008, 11:56 p.m.

897: HP67/97 Navigation Pac I - Star Sight Planner Data Card Update for 2008, MikeO - 11 Aug 2008, 3:45 p.m.

896: HP-35S Multiple cashflow / NPV / IRR / NFV / NUS program, Caspar L - 1 Aug 2008, 2:18 p.m.

893: HP-35S Triangle solve program, Caspar L - 24 July 2008, 4:36 p.m.

887: A Third Order Linear Equation Solver for the HP-35S, Palmer O. Hanson, Jr. - 20 May 2008, 11:24 p.m.

886: An Extension for Stefan's Matrix Multi-Tool Program, Palmer O. Hanson, Jr. - 4 May 2008, 9:24 a.m.

881: A "Cadillac " Quadratic Solver for the hp 33s and HP35s, Palmer O. Hanson, Jr. - 29 Apr 2008, 11:40 a.m.

876: The MoHPC 9114 Repair Manual, PeterP - 26 Apr 2008, 8:37 p.m.

867: HP Calculator Simulators - with downloads, Mike T. - 5 Apr 2008, 5:08 p.m.

866: Effective Computer-aided Calculator Programming - Part 1 - Voyager, Egan Ford - 2 Apr 2008, 12:47 a.m.

836: Eramco Systems RSU-2 Soft Select (128K) for HP-41, Diego Diaz - 1 Apr 2008, 12:41 a.m.

826: Making sense of all the 41c Service ROM versions: 1A 1C 1D 2A 2B, Allen - 10 Mar 2008, 12:04 a.m.

802: 41C to 50g Migration, Egan Ford - 1 Mar 2008, 4:48 a.m.

780: HP-41C RECURSION SOLVES IT, designnut - 25 Feb 2008, 3:54 p.m.

763: Control the World with HP-IL/RS-232/41CX + X10, Egan Ford - 8 Sept 2007, 4:04 p.m.

746: Explore the World with HP-IL/RS-232/71B + GPS, Egan Ford - 8 Sept 2007, 3:32 p.m.

735: HP-35s bug list, Paul Dale - 22 Aug 2007, 6:22 p.m.

731: UserRPL stack commands, James M. Prange (Michigan) - 8 July 2007, 3:51 a.m.

722: A Third Order Complex Linear Equation Solver for the hp 33s, Palmer O. Hanson, Jr. - 20 June 2007, 10:27 p.m.

716: Making replacements for 82001 Classic Battery, Don Shaffer - 20 Apr 2007, 6:01 p.m.

712: programming using the HP 17bii+ solver, Don Shepherd - 14 Apr 2007, 2:48 p.m.

709: HP Calculator Software- Literature Survey, Charles - 31 Mar 2007, 4:38 p.m.

701: EE calculations, some not obvious, Sam Levy - 28 Mar 2007, 10:17 p.m.

700: Calculator Benchmark, Xerxes - 12 Feb 2007, 7:35 a.m.

697: HP-9845 used to produce footage for "Wargames" the movie, John - 6 Feb 2007, 7:16 a.m.

695: Fast and Accurate Trigonometric Functions on the HP-17BII, Gerson W. Barbosa - 12 Jan 2007, 5:17 p.m.

678: Eighth Order Linear Equation Solver for the hp 33s, Palmer O. Hanson, Jr. - 6 Oct 2006, 12:24 a.m.

676: Sixth Order Linear Equation Solver for the hp 33s, Palmer O. Hanson, Jr. - 16 Sept 2006, 10:37 p.m.

654: Fast and Accurate Trigonometric Functions on the HP-12C Platinum, Gerson W. Barbosa - 6 Aug 2006, 3:10 p.m.

601: EMU41 Update 2006, Christoph Klug - 15 June 2006, 4:44 p.m.

599: A Label Saving Simultaneous Equation Solution for the 33s, Palmer O. Hanson, Jr. - 29 May 2006, 10:19 a.m.

583: HP41CY - Replacing the internal +3v lithium cell, Christoph Klug - 27 Feb 2006, 4:09 a.m.

582: Some thoughts about possibilities uploading High-End Pioneer and Clamshell ROM images, Christoph Giesselink - 20 Feb 2006, 3:27 p.m.

580: Replacing the HP-9100 decimal select switch, David Smith - 6 Feb 2006, 7:07 p.m.

578: Repairing 9120 (and possibly other) printer platens, David Smith - 24 Jan 2006, 7:58 p.m.

577: Woodstock RAM chips, Eric Smith - 18 Jan 2006, 5:27 p.m.

- [575: Internet Protocol \(v4\) Addresses on the HP-16C](#), Cameron Paine - 20 Dec 2005, 2:36 a.m.
- [573: E-mails from Everett Kaser on the HP-85](#), john - 19 Dec 2005, 9:29 p.m.
- [570: "Double Memory" for the HP-33s](#), Paul Brogger - 12 Dec 2005, 6:41 p.m.
- [565: HP41C MCODE MOVE](#), Doug Wilder - 5 Nov 2005, 2:19 a.m.
- [563: My 2225B Thinkjet printer battery pack is dead: How can I recell it ?](#), Etienne Victoria - 25 Oct 2005, 5:10 p.m.
- [562: Accessing an HP-9133XV disk drive with an HP-85](#), Juergen (CH) - 12 Oct 2005, 8:56 a.m.
- [560: Transferring files from Series 80 Emulator to HP-85 via floppy disks](#), Juergen (CH) - 12 Oct 2005, 7:52 a.m.
- [556: SOLVE and INTEG on HP's RPN-based models](#), Karl Schneider - 2 Oct 2005, 12:05 a.m.
- [550: HP-IL, Linux and the Internet](#), Howard Owen - 27 July 2005, 4:07 p.m.
- [542: A Unix Geek Rediscovered the HP-41](#), Howard Owen - 5 June 2005, 8:08 p.m.
- [487: Fast and Accurate Trigonometric Functions on the HP-12C](#), Gerson Washiski Barbosa - 22 Apr 2005, 2:33 a.m.
- [484: I have a flashing star \(or snowflake\) on the bottom left of my 1xc's display. What might it be?](#), Etienne Victoria - 14 Apr 2005, 1:19 p.m.
- [483: Frequency](#), Eric Smith - 11 Apr 2005, 4:32 a.m.
- [480: A Narrative History of HP from 1958-1990](#), John Minck - 5 Apr 2005, 4:07 p.m.
- [470: Hyperbolic Approximations for Trigonometric Functions on the HP-12C](#), Gerson Washiski Barbosa - 26 Feb 2005, 1:27 p.m.
- [459: Bad Tones / Good Tones on HP41CX](#), Thomas (CH) - 13 Jan 2005, 12:47 p.m.
- [451: Pioneer keyboard schematic](#), Randy Sloyer - 13 Nov 2004, 8:36 p.m.
- [448: Uploading ROM's for HP41 & EMU41](#), Christoph Klug - 24 Sept 2004, 10:35 a.m.
- [437: Primitive RPN Calculators](#), Palmer O. Hanson, Jr. - 21 Apr 2004, 12:34 a.m.
- [431: "Linear Programming" to Minimize LBL Usage on the HP-33s](#), Paul Brogger - 15 Mar 2004, 8:55 p.m.
- [428: HP-like RPN Java calculator for mobile phones](#), Roar Lauritzsen - 27 Feb 2004, 4:16 a.m.
- [420: homebrew woodstock batt pack](#), kevin clevenger - 27 Feb 2004, 12:49 a.m.
- [416: HP12C Platinum "curiosities"](#), Vieira, Luiz C. (Brazil) - 24 Jan 2004, 9:01 p.m.
- [408: HP-49G+ -- A Look Inside . . .](#), Paul Brogger - 8 Dec 2003, 9:50 p.m.
- [396: Fifteen Digit Precision on the HP-32S](#), Palmer O. Hanson, Jr. - 18 Oct 2003, 12:43 a.m.
- [394: HP 67 Internals](#), Tony Duell - 2 Oct 2003, 9:42 p.m.
- [391: HP Charger Data](#), Tony Duell (UK) - 18 Sept 2003, 9:25 p.m.
- [390: Classic Charger Schematic](#), Tony Duell (UK) - 18 Sept 2003, 9:23 p.m.
- [389: HP 65 Internals](#), Tony Duell (UK) - 18 Sept 2003, 9:22 p.m.
- [388: HP 55 Internals](#), Tony Duell (UK) - 18 Sept 2003, 9:19 p.m.
- [387: HP 45 Internals](#), Tony Duell (UK) - 18 Sept 2003, 9:18 p.m.
- [385: HP 35 internals](#), Tony Duell (UK) - 18 Sept 2003, 9:15 p.m.
- [380: HP-97S Test Connector](#), Erik Wahlin (USA) - 6 Aug 2003, 9:37 p.m.
- [372: HP-15C First Degree Runge-Kutta Order 4](#), Patrick - 4 Aug 2003, 2:56 p.m.
- [361: Service tips for new owners of HP-34C, other spice calc's](#), Norm - 5 June 2003, 9:02 p.m.
- [350: HP16C Trigonometric Functions \(no inverses so far\)](#), Vieira, Luiz C. (Brazil) - 13 May 2003, 1:46 a.m.
- [335: Replacement Nicads for HP Battery Packs](#), Katie Wasserman - 4 Mar 2003, 5:55 p.m.
- [329: Creating HP41 Double X-Memory Modules](#), Christoph Klug - 3 Mar 2003, 5:55 a.m.
- [327: Flatten vintage wrinkled HP card holders](#), Mark Hoskins - 28 Feb 2003, 12:41 p.m.
- [320: Program lists for HP-12C, 15C, 16C](#), Peter Heberer - 24 Feb 2003, 9:46 a.m.
- [313: HP28C: Hardware I/O basics](#), Christoph Giesselink - 20 Jan 2003, 3:30 p.m.
- [311: HP41 - PC Gateway & Emulator](#), Christoph Klug - 7 Jan 2003, 5:44 a.m.
- [310: Hiding Files in HP48G series calculator](#), HPunk® - 5 Jan 2003, 11:12 a.m.
- [305: Solving the problem of "HP49 Crashes when installing libraries in Flash ROM \(Port 2\)"](#), HPunk® - 15 Dec 2002, 8:21 a.m.
- [302: Opening a Voyager](#), Steve - 29 Nov 2002, 4:24 p.m.
- [301: ElectroStatic Discharge: a possible solution for the HP48](#), Vieira, Luiz C. (Brazil) - 27 Nov 2002, 7:25 a.m.
- [293: Removing the back labels to HP classic series calculators](#), Mark Hoskins - 10 Nov 2002, 11:53 p.m.

- 286: True Type Fonts® for the HP41, Voyagers and 82143A**, Vieira, Luiz C. (Brazil) - 25 Aug 2002, 4:16 p.m.
- 281: 99 Digits of PI on an HP 32SII**, Katie Wasserman - 19 June 2002, 11:08 p.m.
- 275: HP28S: Hardware I/O basics**, Christoph Giesselink - 19 June 2002, 12:23 p.m.
- 269: HP-19C printer overhaul discoveries**, Michael Meyer - 27 May 2002, 12:11 a.m.
- 268: Calculator Fiction -- The Mutant HP**, Paul Brogger - 24 May 2002, 11:47 a.m.
- 267: A Common Woodstock Problem and Repair Suggestions**, katie - 16 Apr 2002, 10:14 p.m.
- 266: HP-42S: New Facts**, J-F Garnier (France) - 10 Apr 2002, 4:32 p.m.
- 262: hp-67 / hp-97 card reader repair hints (wheel & clutch)**, Bernd Schmeling - 3 Apr 2002, 5:47 p.m.
- 260: liffile - HP9114 LIF disk image file**, Tony Duell - 20 Mar 2002, 11:55 p.m.
- 247: HP 49g...dedication necessary.**, Josh (USA) - 25 Feb 2002, 1:59 p.m.
- 241: Repairing the HP-67**, Mark Hoskins - 30 Jan 2002, 1:49 p.m.
- 239: HP 41C Card Reader Repair, The Short Fix Approach**, Mark Hoskins - 30 Jan 2002, 1:48 p.m.
- 236: HP 41 display reset.**, m robinso - 30 Dec 2001, 2:02 a.m.
- 234: Financial-Management (and Modified) Internal Rate-of-Return for the HP-17BII and HP-19BII**, W. Bruce Maguire II - 29 Nov 2001, 4:49 p.m.
- 225: HP-12C Trigonometric Functions**, Viktor Toth - 21 Oct 2001, 9:52 p.m.
- 222: Improved TRIG. and INVERSE TRIG. functions for the HP-17BII**, W. B. Maguire II - 26 Sept 2001, 12:40 p.m.
- 210: HP-17BII precision: Epsilon**, W. B. Maguire II - 24 Sept 2001, 4:33 p.m.
- 200: Better trig functions on HP17BII using Maclaurin series**, Michael Blankenship - 10 Sept 2001, 11:35 a.m.
- 199: Increasing HP-42s Memory to 32K**, Paul Brogger - 6 Sept 2001, 7:03 p.m.
- 196: HP-97 printer ribbon repair**, Erik Wahlin - 6 Sept 2001, 1:07 a.m.
- 192: Interfacing IL2000 System to Analogue World**, Klug Christoph - 13 Aug 2001, 2:17 p.m.
- 187: TRIG and inverse TRIG functions on the HP-17bII**, W. Bruce Maguire II - 31 July 2001, 7:13 p.m.
- 184: UPDATE: New batteries in old HPs**, W. Bruce Maguire II - 31 July 2001, 3:34 p.m.
- 181: New batteries in old HPs**, W. Bruce Maguire II - 27 July 2001, 4:24 p.m.
- 180: HP-97 Card Reader Repair**, Mike Davis - 17 July 2001, 6:03 p.m.
- 179: HP-67 Card Reader Repair**, Mike Davis - 17 July 2001, 6:02 p.m.
- 170: Golden Ratio and HP-1xC series**, John Ioannidis - 4 July 2001, 12:22 p.m.
- 154: HP 9810/9820 Card Reader Drive Wheel Repair**, Katie Wasserman - 6 Apr 2001, 4:29 p.m.
- 149: Reseting an HP41**, Steve (Australia) - 24 Mar 2001, 6:47 a.m.
- 146: Batt. Pack for HP 80 & HP 67**, Thomas G. Magee - 9 Mar 2001, 4:17 p.m.
- 144: Impressions on the HP-49G**, Ernie Malaga - 2 Mar 2001, 9:12 a.m.
- 139: HP-41 Fonts**, Ernie Malaga - 27 Feb 2001, 6:09 p.m.
- 138: HP46 Printer Repair Tip**, Alex Knight - 26 Feb 2001, 3:43 p.m.
- 128: Opening the HP48: Addendum**, David Fenyes - 20 Feb 2001, 8:31 p.m.
- 125: HP48 Disassembly (from the back)**, David Fenyes - 18 Feb 2001, 3:00 p.m.
- 118: Serial port downloads from (Linux) PC to HP-41**, Tony Duell - 10 Feb 2001, 2:59 p.m.
- 111: HP 65 card reader tips: reinforcing the worm gear clutch**, Rafa, Spain - 28 Jan 2001, 3:55 p.m.
- 108: The (not so) Big List of HP Part Numbers**, John Kono - 5 Jan 2001, 4:27 a.m.
- 86: HP Calculator Opcode Map**, George Weigt - 14 Dec 2000, 5:48 p.m.
- 84: HP-97 Printer Gear Replacement**, Katie Wasserman - 15 Nov 2000, 2:07 a.m.
- 79: HP-67 automatic documenting program for Excel**, John H. Meyer - 10 Sept 2000, 4:54 p.m.
- 78: HP9114 repairs (not for the faint hearted)**, Steve (Australia) - 9 Sept 2000, 12:40 p.m.
- 74: IR Printer information**, Steve (Australia) - 31 July 2000, 11:00 a.m.
- 71: Some information on the CCD-Rom -W&W CCD A or -W&W CCD B**, Reinhard Breuer - 24 July 2000, 11:10 p.m.
- 63: Repairing pre-recorded cards for HP-65, 67, 97**, Erik Wahlin - 20 July 2000, 8:50 a.m.
- 62: How to un-private an HP-41 program**, Steve (Australia) - 22 June 2000, 6:05 a.m.
- 60: Recovering shorted battery packs**, Steve (Australia) - 19 June 2000, 2:40 a.m.
- 57: Repairing the hp 82104A Card Reader**, Dave the "lostrabbit" - 25 May 2000, 12:05 p.m.
- 54: HP-41/PC User-Code Transfers**, Leo Duran - 24 May 2000, 12:16 p.m.

- 42: HP-41C "Decompiler" and "BarCode Maker"**, Dan McDonald - 14 May 2000, 1:11 p.m.
27: AWK Translator for Keycodes, bill duncan, ve3ied - 23 Apr 2000, 1:08 a.m.
26: The continued fractions algorithm (was: Re: Converting a decimal # to a fraction #), rupert9000 - 22 Apr 2000, 6:09 p.m.
24: HP-IL Files: Where, What, Why, How, etc., Dan McDonald - 7 Apr 2000, 12:34 p.m.
5: Pioneer "Observational Internals", Paul J. Brogger - 2 Apr 2000, 9:29 p.m.
3: HP-41 Hardware Interface Activities, Klug Christoph - 2 Apr 2000, 9:22 p.m.
2: Repairing an HP-41CX, Stefan Vorkoetter - 2 Apr 2000, 9:20 p.m.
1: Hewlett Packard HP-25 Calculator: The Minimum Computer, James Blodgett - 2 Apr 2000, 9:14 p.m.

[[Return to the Message Index](#)]



[Go to the HP Forum archives selection page](#)



[Go back to the main exhibit hall](#)



The Museum of HP Calculators



The Museum of HP Calculators displays and describes Hewlett-Packard calculators introduced from 1968 to 1986 plus a few interesting later models. There are also sections on calculating machines and slide rules as well as sections for buying and selling HP calculators, an HP timeline, collecting information and a software library.

The Main Exhibit Hall

Before HP Calculators (1614-1968)

- [The HP Garage](#) - The Birthplace of Silicon Valley
- [Calculating Machines](#), [Slide Rules](#), etc.

The First HP Calculators (1968)

- [HP 9100A&B](#) Programmable RPN Calculators with *no digital ICs*

High End Desktop Series (1971)

- [HP 9810A](#) Second Generation RPN Calculator
- [HP 9820A](#) Full Algebraic Desktop
- [HP 9830A](#) Desktop Calculator with BASIC
- [HP 9805A](#) Algebraic Statistics Desktop
- [HP 9815A/S](#) Third Generation RPN Desktop
- [HP 9825A](#) First HPL Calculator

The Classics (1972)

- [HP-35](#) The First Slide Rule Pocket Calculator
- [HP-80](#) The First Business Calculator
- [HP-45](#) Slide Rule++
- [HP-65](#) The First Programmable Handheld
- [HP-70](#) Inexpensive Business Calculator
- [HP-55](#) Scientific/Programmable with Quartz Timer

Classic Printing Desktop Calculators (1973)

Navigation

- [What's New?](#) (12/11/13)
- [Search the Museum](#)
- [Manuals on CD/DVD](#)
- [Escape frames](#)
- [Contact the Curator](#)

Discussion & Guest Areas

- [HP Forum](#)
(Includes everything from the earliest HPs to the HP Prime, plus the new articles forum, new software forums and classifieds.)
- [Old HP Forum](#)
- [Old HP Forum Archives](#)
- [Old Articles Forum](#)

Collecting & Using

- [Classified Ads](#)

- [HP-46](#) Printing Version of the HP-45
- [HP-81](#) Printing Version of the HP-80
- [HP-46](#), [56](#), [66](#) and HP-9805 Design Studies.

Second Generation Handheld Calculators (1975)

- [HP-10](#) Handheld Printing Adding Machine
- [HP-21](#) Second Generation HP-35
- [HP-22](#) Business Calculator
- [HP-25](#) Scientific Programmable
- [HP-25C](#) HP-25 with Continuous Memory
- [HP-27](#) "Do Everything" Model (Sci/Stat/Math/Bus)
- [HP-29C & HP-19C](#) High End Programmables

Second Generation Desktop Calculators (1976)

- [HP-91](#) Printing Scientific Calculator
- [HP-92](#) Printing Business/Financial Calculator
- [HP-95C](#) Unreleased Printing Programmable Calculator

Second Generation Mag Card Programmables (1976)

- [HP-67 & HP-97](#) Pocket & Desktop Calculators
- [HP-97S](#) I/O Calculator

In A Class By Itself (1977)

- [HP-01](#) The Hewlett-Packard Calculator Watch

Third Generation of HP Calculators (1978)

- [HP-31E](#) Basic Scientific Calculator
- [HP-32E](#) Scientific/Statistical
- [HP-33E/C](#) Scientific Programmable Calculator
- [HP-34C](#) Scientific Prog. with Integration & Solve
- [HP-37E](#) Financial Calculator
- [HP-38E/C](#) Programmable Financial

Small Computers and Computer-like Calculators (1979)

- [HP-41C](#) Alphanumeric Programmables (includes CV & CX)
- [HP-80 Series](#) Small Computers
- [HP-75C/D](#) Smaller Computer
- [HP-71B](#) Even Smaller Computer/Calculator
- [HP-94D/E/F](#) Handheld Industrial Computer

The 10C series: Compact yet Sophisticated Calculators (1981)

- [HP-10C](#) Basic Scientific Programmable Calculator

- [Collector's Corner](#)
- [Prices & Rarity](#)
- [Series & Code Names](#)
- [Variations & Specials](#)
- [Bugs](#)
- [Battery Packs & Chargers](#)
- [Repairs & Batteries](#)
- [Serial Numbers](#)
- [Self Tests](#)
- [Where To Find](#)
- [Supplies](#)
- [Manuals on CD](#)
- [Learning RPN](#)
- [Programming](#)
- [Calculator Simulations](#)
- [Compare Features](#)
- [Adding a crystal to the HP-45](#)
- [FAQ](#)

Software Libraries

- [New software Libraries](#)
(Software submitted after Dec 2013 is here.)
- [Old Software Library](#)
- [Old HP-41C Software Library](#)
- [HP-95C Software Library](#)
- [Software on CD](#)

History & Technology

- [A brief time line](#)
- [Calculator Technologies](#)
- [CPUs & Internal Programming](#)
- [Benchmarks](#)
- [Compare Features](#)
- [HP Journal Library](#)
- [Hewlett Packard Advertisements](#)
- [HP Manuals on CD](#)
- [RPN Versions](#)
- [Photo Gallery](#)

For Sale and Wanted

- [Items Wanted](#)
 - [Classified Ads](#)
-

- [HP-11C](#) More Advanced Scientific Programmable
- [HP-12C](#) Top Selling Business Calculator
- [HP-15C](#) Scientific with Matrix & Complex Math
- [HP-16C](#) The Computer Scientist Calculator

The RPL Family (1986)

- [RPL](#) A Platform for Powerful Calculators
- [HP-18C](#) Business Model with Solver
- [HP-28C/S](#) Scientifics with Symbolic Math and Solver
- [HP-48S/SX](#) Large Screen Scientifics with Symbolic Math

Some Interesting Later Models (1986)

- [HP-27S](#) Algebraic Do Everything Calculator
- [HP-17B](#) Algebraic Business Model
- [HP-22S](#) Algebraic Scientific Calculator
- [HP-32S](#) RPN Scientific with Complex Math (Including Anniversary Edition.)
- [HP-42S](#) RPN Scientific (HP-41C compatible)
- [HP-14B](#) (50th Anniversary Edition)
- [The Xpander](#) Unreleased Math Learning Aid
- [HP-67CX](#) Unreleased RPN / HPL / Linux Scientific
- [Other](#) Hewlett Packard Calculators and computers.

Other Resources

- [Links to other Sites](#)
- [Books and Magazines](#)
- [Associations](#)
- [The Museum on CD](#)

The Museum of HP Calculators and the museum curator are not affiliated with the Hewlett-Packard Company.

Text and images are Copyright [David G. Hicks](#), 1995 - 2013. Quotations of HP manuals are Copyright Hewlett Packard Company - various years. For questions about me and the origin of the museum, please see my [personal page](#).

[Contact David Hicks](#)



The Museum of HP Calculators



New Article in Old Forum

Please note that this is an old forum. Please start new threads on the the [NEW HP Forum](#)

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



[[Return to the Index](#)]

HP Articles Forum

Message Index Display Configuration

Use the form below to search for messages which include certain keywords or which were posted at a particular time or by a particular individual, and/or to select the manner in which you wish the messages in the index to be displayed.

List messages

A chronological display will put the newest messages at the top of the list. A standard threaded display will put the newest messages at the bottom. A reversed threaded display will show messages in a mixed order: 'First' messages will be arranged with the newest at the top, but responses will be arranged with the newest at the bottom. Though awkward, this is the default index style of many Web-based bulletin boards, and is the style with which many users are most familiar. A compressed listing will show on the main index page only the first message of each thread.)

List only those messages:

☒ Posted within the last

(Options above this line are "sticky" if your browser accepts cookies. Options below are for the next view of the index only.)

☐ Posted between and

List:

☒ All messages within the specified date range

☐ Only messages containing of the following keywords:

☐ Only messages posted by:

[View Message Index](#)

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[New Article](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
 [[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) |
[Ads without Auctions](#)]

Welcome back!
 Your last visit began on 30 Oct 2014, 10:39 p.m.
 Since then, no new articles have been posted.

This forum is intended to allow users to contribute more comprehensive articles than those that are typically found in the general forums. This could include information such as advanced programming and repair instructions as well thoughts about specific calculators or HP calculators in general. Please choose your titles carefully so your information can be easily found.

Please do **not** ask questions here - it is not possible to reply to the messages in this forum. If you want to refer to one of these articles in the general forum, please refer to it by name or article number Please use the [HP Calculator Ads](#) section for "for sale" and "wanted" ads. Please review the [Terms of Use](#) for more complete information.

It's possible to use some [advanced formatting techniques](#) in this forum if you wish.

Article Index

Articles Posted or Modified Within the Last Archive

*210 of 210 Articles Displayed
 (Reversed Threaded Listing)*

- [1266: HP-41 MCODE: The Last Function - at last!](#), Ángel Martin - 8 Nov 2013, 4:55 a.m.
- [1263: HP-80 History, Design and Interesting Facts](#), Max Stone - 24 Oct 2013, 4:14 p.m.
- [1253: RPN-1200](#), Benoit Maag - 25 Aug 2013, 1:02 a.m.
- [1252: 41 SOLVE & INTEG Bank-switched Implementation.](#), Ángel Martin - 21 June 2013, 11:12 a.m.
- [1246: Computing Sunrise/Sunset with the HP 35s](#), Justin Sloan (Honolulu) - 29 Apr 2013, 12:30 a.m.
- [1240: HP-42S with Electroluminescent screen and simple I/O port](#), Jose Poyan - 26 Mar 2013, 4:45 p.m.
- [1220: Sudoku Solver for the HP-15c](#), Marcel Samek - 4 Feb 2013, 1:20 p.m.
- [1218: Mapping HP48 Text to Unicode](#), Chris Dreher - 16 Jan 2013, 1:15 a.m.
- [1205: HP-41 "Halfnut": patching the ROM for full lower case character set display](#), J-F Garnier - 28 Dec 2012, 12:25 p.m.
- [1200: 41-CL Routine of the week: MMUSWAP](#), Ángel Martin - 4 Nov 2012, 5:57 a.m.
- [1197: Palmer's Matrix Multi-Tool Extension](#), mbrethen - 30 Sept 2012, 4:43 p.m.
- [1196: HP-UX and RPL/2](#), BERTRAND Joël - 26 Sept 2012, 6:03 a.m.
- [1195: 41-MCODE: Breaking the FAT Barrier](#), Ángel Martin - 3 Sept 2012, 6:16 a.m.
- [1183: Make PIL-Box ILPer program run on OS X](#), Juergen Keller - 2 Sept 2012, 4:24 p.m.
- [1179: HP-82163A how-to and video](#), Sylvain Cote - 21 July 2012, 2:47 p.m.
- [1174: HP-41 Post Replacement](#), aj04062 - 12 July 2012, 9:45 p.m.
- [1173: 41-MCODE: Dr. Jekyll & Mr. Hyde](#), Ángel Martin - 7 July 2012, 3:48 a.m.
- [1172: Calculation of factorials using logarithms \(HP41 MCODE\)](#), Frido Bohn - 26 May 2012, 9:15 a.m.
- [1162: 41CL - port configuration](#), Christoph Klug - 22 Mar 2012, 5:14 p.m.
- [1158: HP-41 MCODE: Checking ROMs Configuration](#), Ángel Martin - 18 Mar 2012, 3:23 a.m.

- 1152: HP 17BII+ TVM Solver Solution expressing the Balloon Payment as a Percentage with Begin and End mode [and Down Payment as a percentage]**, bmaskell - 3 Mar 2012, 12:36 a.m.
- 1150: Resurrecting a shattered HP 19B**, Neil Hamilton (Ottawa) - 30 Jan 2012, 2:14 p.m.
- 1149: HP41 Trip Computer**, Christopher Johnson - 29 Jan 2012, 9:36 p.m.
- 1133: [WP34s] The ultimate Polynomial RootSolver for the WP34s**, fhub - 3 Jan 2012, 7:23 a.m.
- 1131: RPN Sunrise / Sunset programs for HP41C/HP42S/HP33S/HP35S**, M. Joury - 14 Dec 2011, 9:25 a.m.
- 1129: Incomplete Gamma Function for wp34s**, Les Wright - 5 Dec 2011, 2:40 a.m.
- 1126: Romberg Integration on the wp34s**, Les Wright - 4 Dec 2011, 7:43 p.m.
- 1118: WP 34S: Polynomial root finder adapted from Valentin Albillo's HP 41 program**, Miguel Toro - 23 Nov 2011, 7:32 p.m.
- 1117: WP 34S: financial routines IRR, NPV, Amortization, Interest rate conversion (updated 2013-04-28)**, Miguel Toro - 12 Nov 2011, 2:20 p.m.
- 1113: Inside Saturn Emulator Timing**, Christoph Giesselink - 20 Oct 2011, 4:56 p.m.
- 1107: HP 15 LE: Errata list for Owner's Handbook**, Felix Gross - 10 Oct 2011, 3:03 a.m.
- 1104: WP 34S: Integration wrapper**, Miguel Toro - 9 Oct 2011, 10:26 p.m.
- 1102: WP 34S: MIO matrix helper * edited***, Miguel Toro - 8 Oct 2011, 8:53 p.m.
- 1096: HP35s date functions: DOW, DDate, Date+, Gregorian to Julian conversions**, David Persinger (US) - 16 Sept 2011, 2:21 p.m.
- 1089: HP-15C LE (Limted Edition) Bug Reports**, Katie Wasserman - 16 Sept 2011, 11:22 a.m.
- 1088: Simple TVM Routines for HP-15C**, Chris McCormack - 16 Sept 2011, 8:46 a.m.
- 1072: Creating ROM image files of an HP-41 module**, Juergen Keller - 9 Aug 2011, 4:55 p.m.
- 1062: Thank You HP Users**, LouisD - 9 Aug 2011, 3:21 p.m.
- 1048: EE calculations, some not obvious (reformatted)**, Thomas Klemm - 26 May 2011, 5:14 p.m.
- 1038: Multiple Period Percentage Change Program for the 30b**, Mark Harman - 25 May 2011, 12:13 a.m.
- 1037: HP 30b programs for interest rate conversions with negative or fractional periods per year**, gene wright - 3 May 2011, 2:30 p.m.
- 1035: Game for HP-35s**, Paul Dale - 16 Mar 2011, 5:46 a.m.
- 1025: N-queens benchmark and the HP-17BII**, Thomas Klemm - 7 Feb 2011, 5:48 a.m.
- 1024: Sum of Digits Divisibility Challenge - an unusual solution**, David Hayden - 16 Jan 2011, 9:39 p.m.
- 1021: How to use a formula in a post**, Thomas Klemm - 8 Nov 2010, 4:11 p.m.
- 1014: Sunrise and Sunset**, Thomas Klemm - 28 Oct 2010, 12:05 p.m.
- 1003: HP 35s. Quadratic Equation Solver (with complex solutions support)**, Pablo P (Spain) - 2 July 2010, 3:53 p.m.
- 1002: Calculator Benchmark: Loops of addition**, gene wright - 2 June 2010, 2:51 p.m.
- 1001: The Caves - Reloaded**, Frido Bohn - 6 May 2010, 5:03 a.m.
- 991: Table of HP48 serial numbers and ROM versions**, megarat - 13 Apr 2010, 12:22 p.m.
- 990: HP 82490A Translator Pac Overlay**, Allen - 8 Feb 2010, 11:32 a.m.
- 989: R->P and P->R for the 35s, Reloaded**, Frido Bohn - 26 Jan 2010, 4:06 a.m.
- 986: Psychrometric calculations on the HP 35s (and 50g)**, Maarten Ambaum (Reading, UK) - 22 Jan 2010, 11:14 a.m.
- 983: R->P & P->R for the 35s**, Paul Dale - 21 Jan 2010, 5:22 a.m.
- 982: nCr and nPr - a 41-MCODE implementation**, Ángel Martin - 27 Nov 2009, 10:46 a.m.
- 980: Fast DiGamma Function for Positive and Negative Real Arguments**, Steven Thomas Smith - 26 Sept 2009, 6:06 a.m.
- 978: Fast Incomplete Gamma/Exponential Integral/Error Functions With Lentz's Method**, Steven Thomas Smith - 23 Sept 2009, 1:28 p.m.
- 944: EIGEN: Characteristic equation+eigenvalues for order 2,3,4,5**, Steven Thomas Smith - 1 Sept 2009, 12:39 p.m.
- 941: GAMMA and GAMMLN: Fast, Accurate Lanczos Approximation for Gamma Functions**, Steven Thomas Smith - 1 Sept 2009, 12:49 p.m.
- 937: Enhance your HP-97 -- Add a Clock and More**, Katie Wasserman - 1 Sept 2009, 6:53 p.m.
- 928: HP41 Prime Factorization Revisited**, Ángel Martin - 27 July 2009, 3:58 a.m.
- 926: Mental juggling: Calculating the day of week**, Thomas Klemm - 7 July 2009, 1:36 p.m.
- 925: A non destructive (easy) method for opening a spice calculator.**, Geoff Quickfall - 27 May 2009, 1:47

p.m.

923: HP 41 Battery posts, another repair, Geoff Quickfall - 20 Apr 2009, 2:58 p.m.

910: Power for HP82162A Printer and HP9114 Disc Drive, Philip Isaac (Canada) - 3 Mar 2009, 12:02 p.m.

909: INIT-Module for ERAMCO RSU-2 Rambox, PeterP - 18 Feb 2009, 6:37 p.m.

908: HP-71B Math RAM, Marcus von Cube, Germany - 16 Feb 2009, 8:06 a.m.

903: HP41 - How to mix and match UCODE and MCODE in a module, PeterP - 13 Feb 2009, 7:37 p.m.

902: cheaper batteries for the HP41CX, Michael Harwood - 30 Jan 2009, 9:32 p.m.

901: HP9815 - repairing the paper advance mechanism, Mark Edmonds - 19 Oct 2008, 8:29 a.m.

899: Many Digits of Pi (HP32sii, HP-16c, HP-12c, HP-12cp, HP-19c, HP-30b, HP-15c), Katie Wasserman - 17 Oct 2008, 11:56 p.m.

897: HP67/97 Navigation Pac I - Star Sight Planner Data Card Update for 2008, MikeO - 11 Aug 2008, 3:45 p.m.

896: HP-35S Multiple cashflow / NPV / IRR / NFV / NUS program, Caspar L - 1 Aug 2008, 2:18 p.m.

893: HP-35S Triangle solve program, Caspar L - 24 July 2008, 4:36 p.m.

887: A Third Order Linear Equation Solver for the HP-35S, Palmer O. Hanson, Jr. - 20 May 2008, 11:24 p.m.

886: An Extension for Stefan's Matrix Multi-Tool Program, Palmer O. Hanson, Jr. - 4 May 2008, 9:24 a.m.

881: A "Cadillac " Quadratic Solver for the hp 33s and HP35s, Palmer O. Hanson, Jr. - 29 Apr 2008, 11:40 a.m.

876: The MoHPC 9114 Repair Manual, PeterP - 26 Apr 2008, 8:37 p.m.

867: HP Calculator Simulators - with downloads, Mike T. - 5 Apr 2008, 5:08 p.m.

866: Effective Computer-aided Calculator Programming - Part 1 - Voyager, Egan Ford - 2 Apr 2008, 12:47 a.m.

836: Eramco Systems RSU-2 Soft Select (128K) for HP-41, Diego Diaz - 1 Apr 2008, 12:41 a.m.

826: Making sense of all the 41c Service ROM versions: 1A 1C 1D 2A 2B, Allen - 10 Mar 2008, 12:04 a.m.

802: 41C to 50g Migration, Egan Ford - 1 Mar 2008, 4:48 a.m.

780: HP-41C RECURSION SOLVES IT, designnut - 25 Feb 2008, 3:54 p.m.

763: Control the World with HP-IL/RS-232/41CX + X10, Egan Ford - 8 Sept 2007, 4:04 p.m.

746: Explore the World with HP-IL/RS-232/71B + GPS, Egan Ford - 8 Sept 2007, 3:32 p.m.

735: HP-35s bug list, Paul Dale - 22 Aug 2007, 6:22 p.m.

731: UserRPL stack commands, James M. Prange (Michigan) - 8 July 2007, 3:51 a.m.

722: A Third Order Complex Linear Equation Solver for the hp 33s, Palmer O. Hanson, Jr. - 20 June 2007, 10:27 p.m.

716: Making replacements for 82001 Classic Battery, Don Shaffer - 20 Apr 2007, 6:01 p.m.

712: programming using the HP 17bii+ solver, Don Shepherd - 14 Apr 2007, 2:48 p.m.

709: HP Calculator Software- Literature Survey, Charles - 31 Mar 2007, 4:38 p.m.

701: EE calculations, some not obvious, Sam Levy - 28 Mar 2007, 10:17 p.m.

700: Calculator Benchmark, Xerxes - 12 Feb 2007, 7:35 a.m.

697: HP-9845 used to produce footage for "Wargames" the movie, John - 6 Feb 2007, 7:16 a.m.

695: Fast and Accurate Trigonometric Functions on the HP-17BII, Gerson W. Barbosa - 12 Jan 2007, 5:17 p.m.

678: Eighth Order Linear Equation Solver for the hp 33s, Palmer O. Hanson, Jr. - 6 Oct 2006, 12:24 a.m.

676: Sixth Order Linear Equation Solver for the hp 33s, Palmer O. Hanson, Jr. - 16 Sept 2006, 10:37 p.m.

654: Fast and Accurate Trigonometric Functions on the HP-12C Platinum, Gerson W. Barbosa - 6 Aug 2006, 3:10 p.m.

601: EMU41 Update 2006, Christoph Klug - 15 June 2006, 4:44 p.m.

599: A Label Saving Simultaneous Equation Solution for the 33s, Palmer O. Hanson, Jr. - 29 May 2006, 10:19 a.m.

583: HP41CY - Replacing the internal +3v lithium cell, Christoph Klug - 27 Feb 2006, 4:09 a.m.

582: Some thoughts about possibilities uploading High-End Pioneer and Clamshell ROM images, Christoph Giesselink - 20 Feb 2006, 3:27 p.m.

580: Replacing the HP-9100 decimal select switch, David Smith - 6 Feb 2006, 7:07 p.m.

578: Repairing 9120 (and possibly other) printer platens, David Smith - 24 Jan 2006, 7:58 p.m.

577: Woodstock RAM chips, Eric Smith - 18 Jan 2006, 5:27 p.m.

- [575: Internet Protocol \(v4\) Addresses on the HP-16C](#), Cameron Paine - 20 Dec 2005, 2:36 a.m.
- [573: E-mails from Everett Kaser on the HP-85](#), john - 19 Dec 2005, 9:29 p.m.
- [570: "Double Memory" for the HP-33s](#), Paul Brogger - 12 Dec 2005, 6:41 p.m.
- [565: HP41C MCODE MOVE](#), Doug Wilder - 5 Nov 2005, 2:19 a.m.
- [563: My 2225B Thinkjet printer battery pack is dead: How can I recell it ?](#), Etienne Victoria - 25 Oct 2005, 5:10 p.m.
- [562: Accessing an HP-9133XV disk drive with an HP-85](#), Juergen (CH) - 12 Oct 2005, 8:56 a.m.
- [560: Transferring files from Series 80 Emulator to HP-85 via floppy disks](#), Juergen (CH) - 12 Oct 2005, 7:52 a.m.
- [556: SOLVE and INTEG on HP's RPN-based models](#), Karl Schneider - 2 Oct 2005, 12:05 a.m.
- [550: HP-IL, Linux and the Internet](#), Howard Owen - 27 July 2005, 4:07 p.m.
- [542: A Unix Geek Rediscovered the HP-41](#), Howard Owen - 5 June 2005, 8:08 p.m.
- [487: Fast and Accurate Trigonometric Functions on the HP-12C](#), Gerson Washiski Barbosa - 22 Apr 2005, 2:33 a.m.
- [484: I have a flashing star \(or snowflake\) on the bottom left of my 1xc's display. What might it be?](#), Etienne Victoria - 14 Apr 2005, 1:19 p.m.
- [483: Frequency](#), Eric Smith - 11 Apr 2005, 4:32 a.m.
- [480: A Narrative History of HP from 1958-1990](#), John Minck - 5 Apr 2005, 4:07 p.m.
- [470: Hyperbolic Approximations for Trigonometric Functions on the HP-12C](#), Gerson Washiski Barbosa - 26 Feb 2005, 1:27 p.m.
- [459: Bad Tones / Good Tones on HP41CX](#), Thomas (CH) - 13 Jan 2005, 12:47 p.m.
- [451: Pioneer keyboard schematic](#), Randy Sloyer - 13 Nov 2004, 8:36 p.m.
- [448: Uploading ROM's for HP41 & EMU41](#), Christoph Klug - 24 Sept 2004, 10:35 a.m.
- [437: Primitive RPN Calculators](#), Palmer O. Hanson, Jr. - 21 Apr 2004, 12:34 a.m.
- [431: "Linear Programming" to Minimize LBL Usage on the HP-33s](#), Paul Brogger - 15 Mar 2004, 8:55 p.m.
- [428: HP-like RPN Java calculator for mobile phones](#), Roar Lauritzsen - 27 Feb 2004, 4:16 a.m.
- [420: homebrew woodstock batt pack](#), kevin clevenger - 27 Feb 2004, 12:49 a.m.
- [416: HP12C Platinum "curiosities"](#), Vieira, Luiz C. (Brazil) - 24 Jan 2004, 9:01 p.m.
- [408: HP-49G+ -- A Look Inside . . .](#), Paul Brogger - 8 Dec 2003, 9:50 p.m.
- [396: Fifteen Digit Precision on the HP-32S](#), Palmer O. Hanson, Jr. - 18 Oct 2003, 12:43 a.m.
- [394: HP 67 Internals](#), Tony Duell - 2 Oct 2003, 9:42 p.m.
- [391: HP Charger Data](#), Tony Duell (UK) - 18 Sept 2003, 9:25 p.m.
- [390: Classic Charger Schematic](#), Tony Duell (UK) - 18 Sept 2003, 9:23 p.m.
- [389: HP 65 Internals](#), Tony Duell (UK) - 18 Sept 2003, 9:22 p.m.
- [388: HP 55 Internals](#), Tony Duell (UK) - 18 Sept 2003, 9:19 p.m.
- [387: HP 45 Internals](#), Tony Duell (UK) - 18 Sept 2003, 9:18 p.m.
- [385: HP 35 internals](#), Tony Duell (UK) - 18 Sept 2003, 9:15 p.m.
- [380: HP-97S Test Connector](#), Erik Wahlin (USA) - 6 Aug 2003, 9:37 p.m.
- [372: HP-15C First Degree Runge-Kutta Order 4](#), Patrick - 4 Aug 2003, 2:56 p.m.
- [361: Service tips for new owners of HP-34C, other spice calc's](#), Norm - 5 June 2003, 9:02 p.m.
- [350: HP16C Trigonometric Functions \(no inverses so far\)](#), Vieira, Luiz C. (Brazil) - 13 May 2003, 1:46 a.m.
- [335: Replacement Nicads for HP Battery Packs](#), Katie Wasserman - 4 Mar 2003, 5:55 p.m.
- [329: Creating HP41 Double X-Memory Modules](#), Christoph Klug - 3 Mar 2003, 5:55 a.m.
- [327: Flatten vintage wrinkled HP card holders](#), Mark Hoskins - 28 Feb 2003, 12:41 p.m.
- [320: Program lists for HP-12C, 15C, 16C](#), Peter Heberer - 24 Feb 2003, 9:46 a.m.
- [313: HP28C: Hardware I/O basics](#), Christoph Giesselink - 20 Jan 2003, 3:30 p.m.
- [311: HP41 - PC Gateway & Emulator](#), Christoph Klug - 7 Jan 2003, 5:44 a.m.
- [310: Hiding Files in HP48G series calculator](#), HPunk® - 5 Jan 2003, 11:12 a.m.
- [305: Solving the problem of "HP49 Crashes when installing libraries in Flash ROM \(Port 2\)"](#), HPunk® - 15 Dec 2002, 8:21 a.m.
- [302: Opening a Voyager](#), Steve - 29 Nov 2002, 4:24 p.m.
- [301: ElectroStatic Discharge: a possible solution for the HP48](#), Vieira, Luiz C. (Brazil) - 27 Nov 2002, 7:25 a.m.
- [293: Removing the back labels to HP classic series calculators](#), Mark Hoskins - 10 Nov 2002, 11:53 p.m.

- [286: True Type Fonts® for the HP41, Voyagers and 82143A](#), Vieira, Luiz C. (Brazil) - 25 Aug 2002, 4:16 p.m.
- [281: 99 Digits of PI on an HP 32SII](#), Katie Wasserman - 19 June 2002, 11:08 p.m.
- [275: HP28S: Hardware I/O basics](#), Christoph Giesselink - 19 June 2002, 12:23 p.m.
- [269: HP-19C printer overhaul discoveries](#), Michael Meyer - 27 May 2002, 12:11 a.m.
- [268: Calculator Fiction -- The Mutant HP](#), Paul Brogger - 24 May 2002, 11:47 a.m.
- [267: A Common Woodstock Problem and Repair Suggestions](#), katie - 16 Apr 2002, 10:14 p.m.
- [266: HP-42S: New Facts](#), J-F Garnier (France) - 10 Apr 2002, 4:32 p.m.
- [262: hp-67 / hp-97 card reader repair hints \(wheel & clutch\)](#), Bernd Schmeling - 3 Apr 2002, 5:47 p.m.
- [260: liffile - HP9114 LIF disk image file](#), Tony Duell - 20 Mar 2002, 11:55 p.m.
- [247: HP 49g...dedication necessary.](#), Josh (USA) - 25 Feb 2002, 1:59 p.m.
- [241: Repairing the HP-67](#), Mark Hoskins - 30 Jan 2002, 1:49 p.m.
- [239: HP 41C Card Reader Repair, The Short Fix Approach](#), Mark Hoskins - 30 Jan 2002, 1:48 p.m.
- [236: HP 41 display reset.](#), m robinso - 30 Dec 2001, 2:02 a.m.
- [234: Financial-Management \(and Modified\) Internal Rate-of-Return for the HP-17BII and HP-19BII](#), W. Bruce Maguire II - 29 Nov 2001, 4:49 p.m.
- [225: HP-12C Trigonometric Functions](#), Viktor Toth - 21 Oct 2001, 9:52 p.m.
- [222: Improved TRIG. and INVERSE TRIG. functions for the HP-17BII](#), W. B. Maguire II - 26 Sept 2001, 12:40 p.m.
- [210: HP-17BII precision: Epsilon](#), W. B. Maguire II - 24 Sept 2001, 4:33 p.m.
- [200: Better trig functions on HP17BII using Maclaurin series](#), Michael Blankenship - 10 Sept 2001, 11:35 a.m.
- [199: Increasing HP-42s Memory to 32K](#), Paul Brogger - 6 Sept 2001, 7:03 p.m.
- [196: HP-97 printer ribbon repair](#), Erik Wahlin - 6 Sept 2001, 1:07 a.m.
- [192: Interfacing IL2000 System to Analogue World](#), Klug Christoph - 13 Aug 2001, 2:17 p.m.
- [187: TRIG and inverse TRIG functions on the HP-17bII](#), W. Bruce Maguire II - 31 July 2001, 7:13 p.m.
- [184: UPDATE: New batteries in old HPs](#), W. Bruce Maguire II - 31 July 2001, 3:34 p.m.
- [181: New batteries in old HPs](#), W. Bruce Maguire II - 27 July 2001, 4:24 p.m.
- [180: HP-97 Card Reader Repair](#), Mike Davis - 17 July 2001, 6:03 p.m.
- [179: HP-67 Card Reader Repair](#), Mike Davis - 17 July 2001, 6:02 p.m.
- [170: Golden Ratio and HP-1xC series](#), John Ioannidis - 4 July 2001, 12:22 p.m.
- [154: HP 9810/9820 Card Reader Drive Wheel Repair](#), Katie Wasserman - 6 Apr 2001, 4:29 p.m.
- [149: Reseting an HP41](#), Steve (Australia) - 24 Mar 2001, 6:47 a.m.
- [146: Batt. Pack for HP 80 & HP 67](#), Thomas G. Magee - 9 Mar 2001, 4:17 p.m.
- [144: Impressions on the HP-49G](#), Ernie Malaga - 2 Mar 2001, 9:12 a.m.
- [139: HP-41 Fonts](#), Ernie Malaga - 27 Feb 2001, 6:09 p.m.
- [138: HP46 Printer Repair Tip](#), Alex Knight - 26 Feb 2001, 3:43 p.m.
- [128: Opening the HP48: Addendum](#), David Fenyes - 20 Feb 2001, 8:31 p.m.
- [125: HP48 Disassembly \(from the back\)](#), David Fenyes - 18 Feb 2001, 3:00 p.m.
- [118: Serial port downloads from \(Linux\) PC to HP-41](#), Tony Duell - 10 Feb 2001, 2:59 p.m.
- [111: HP 65 card reader tips: reinforcing the worm gear clutch](#), Rafa, Spain - 28 Jan 2001, 3:55 p.m.
- [108: The \(not so\) Big List of HP Part Numbers](#), John Kono - 5 Jan 2001, 4:27 a.m.
- [86: HP Calculator Opcode Map](#), George Weigt - 14 Dec 2000, 5:48 p.m.
- [84: HP-97 Printer Gear Replacement](#), Katie Wasserman - 15 Nov 2000, 2:07 a.m.
- [79: HP-67 automatic documenting program for Excel](#), John H. Meyer - 10 Sept 2000, 4:54 p.m.
- [78: HP9114 repairs \(not for the faint hearted\)](#), Steve (Australia) - 9 Sept 2000, 12:40 p.m.
- [74: IR Printer information](#), Steve (Australia) - 31 July 2000, 11:00 a.m.
- [71: Some information on the CCD-Rom -W&W CCD A or -W&W CCD B](#), Reinhard Breuer - 24 July 2000, 11:10 p.m.
- [63: Repairing pre-recorded cards for HP-65, 67, 97](#), Erik Wahlin - 20 July 2000, 8:50 a.m.
- [62: How to un-private an HP-41 program](#), Steve (Australia) - 22 June 2000, 6:05 a.m.
- [60: Recovering shorted battery packs](#), Steve (Australia) - 19 June 2000, 2:40 a.m.
- [57: Repairing the hp 82104A Card Reader](#), Dave the "lostrabbit" - 25 May 2000, 12:05 p.m.
- [54: HP-41/PC User-Code Transfers](#), Leo Duran - 24 May 2000, 12:16 p.m.

- 42: HP-41C "Decompiler" and "BarCode Maker"**, Dan McDonald - 14 May 2000, 1:11 p.m.
27: AWK Translator for Keycodes, bill duncan, ve3ied - 23 Apr 2000, 1:08 a.m.
26: The continued fractions algorithm (was: Re: Converting a decimal # to a fraction #), rupert9000 - 22 Apr 2000, 6:09 p.m.
24: HP-IL Files: Where, What, Why, How, etc., Dan McDonald - 7 Apr 2000, 12:34 p.m.
5: Pioneer "Observational Internals", Paul J. Brogger - 2 Apr 2000, 9:29 p.m.
3: HP-41 Hardware Interface Activities, Klug Christoph - 2 Apr 2000, 9:22 p.m.
2: Repairing an HP-41CX, Stefan Vorkoetter - 2 Apr 2000, 9:20 p.m.
1: Hewlett Packard HP-25 Calculator: The Minimum Computer, James Blodgett - 2 Apr 2000, 9:14 p.m.

[[Return to the Message Index](#)]



[Go to the HP Forum archives selection page](#)



[Go back to the main exhibit hall](#)



[[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) | [Ads without Auctions](#)]

Account Management / Login

Accounts are currently **Required**. You must have an account to post.

Name:

Password:

Login

Login works by setting a cookie. Please make sure your cookies are enabled. If your cookies are disabled, You will need to enter your name and password on each post.

[Create an Account](#)

[I forgot my Password](#)

[I didn't receive my Activation Email](#)

[Accounts Users Guide](#)

[[Account Management / Login](#)]



[Go back to the main exhibit hall](#)



OLD HP Forum

[[New Topic](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
[[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) |
[Ads without Auctions](#)]

Welcome back!

Your last visit began on 30 Oct 2014, 10:34 p.m.

Since then, no new messages have been posted.

This is the OLD HP forum. For newer discussions, please visit the [New HP Forums](#).

How to register at the new forum:

Even if you had an account on the old forum, you'll need to create one for the new forum. In the upper left near "Hello There, Guest" you should see a register link.

Using the same name (or close) as as you used on the old forum is preferred. The new forum does not allow commas in names. If you had a comma before, please omit it from your new account name.

Message Index

Messages Posted or Modified Within the Last Several Eons

(Compressed "Threads Only" Listing)

[Photo of my new Prime](#), Eric Smith - 15 Dec 2013, 2:10 a.m. **14 responses**

[And the collection grows...](#), Les Koller - 14 Dec 2013, 11:29 p.m. **1 response**

[Prime Emulator Connection Problem](#), John Colvin - 14 Dec 2013, 7:18 p.m. **3 responses**

[HP Prime Emulator download](#), John Colvin - 13 Dec 2013, 11:25 p.m. **2 responses**

[HP Prime: SVD](#), JeffG - 13 Dec 2013, 1:47 p.m. **2 responses**

[HP35s Program Four Slings Lift Calculation](#), Jean-Marc Biram (Australia) - 13 Dec 2013, 2:49 a.m. **2 responses**

[Does the HP Prime really compiles the user programs?](#), CompSystems - 13 Dec 2013, 7:22 a.m. **3 responses**

[\[HP-PIRME\] BUG Pretty Print, Solution: abs => |...|, ABS => ||...||](#), CompSystems - 12 Dec 2013, 10:40 p.m. **2 responses**

[An amazing day: Giving a talk at HP about their calculators](#), Geir Isene - 12 Dec 2013, 5:01 p.m. **9 responses**

[Four-Level RPN and the Real World](#), Matt Agajanian - 12 Dec 2013, 4:46 p.m. **14 responses**

[Last call for 41Z/SandMath Overlays... plus new ones](#), Ángel Martin - 12 Dec 2013, 10:27 a.m. **0 responses**

[OT: National Semiconductors NSB-3881](#), Ignazio Cara (Italy) - 12 Dec 2013, 8:31 a.m. **2 responses**

[HP Prime: Baker's Game](#), Mark Power - 11 Dec 2013, 6:25 p.m. **0 responses**

[Reasonable price for a 41C, Quad Mem, Time, Navigation and Card reader module?](#), Eelco Rouw - 11 Dec 2013, 3:15 p.m. **10 responses**

[How to copy files from 9114B to hard drive via PIL-box?](#), Joe Horn - 11 Dec 2013, 12:37 p.m. **23 responses**

[\[HP-PIRME\] ^ OPERTOR // => SUPER BUG](#), CompSystems - 11 Dec 2013, 12:23 p.m. **9 responses**

[The new HP forums](#), Dave Hicks - 11 Dec 2013, 12:12 p.m. **54 responses**

[HP Prime: CAS taylor](#), Alberto Candel - 10 Dec 2013, 7:16 p.m. **5 responses**

[HP50g: Writing a function that returns a function](#), Chris de Castro - 10 Dec 2013, 6:18 p.m. **2 responses**

[Broken keys on HP Prime?](#), Tarcisi C - 10 Dec 2013, 5:59 p.m. **0 responses**

[HP Prime - usbtool.exe](#), bluesun08 - 10 Dec 2013, 10:47 a.m. **21 responses**

[42S with nonvolatile RAM](#), Lyuka - 10 Dec 2013, 9:56 a.m. **5 responses**

[Re: HP Prime: Quite Happy](#), Eddie W. Shore - 10 Dec 2013, 8:19 a.m. **0 responses**

[Re: F1227A as HP48 case: yes!](#), Paul Dale - 10 Dec 2013, 1:01 a.m. **3 responses**

[MINESWEEPER for the HP PRIME](#), Tony Gallo - 10 Dec 2013, 12:26 a.m. **15 responses**

[HP Prime Manual Revisions?](#), Craig Thomas - 9 Dec 2013, 11:08 p.m. **0 responses**

[Anyone here tinker with Arduino/Printbots/etc ?](#), Chris Pem10 - 9 Dec 2013, 8:59 p.m. **3 responses**

[Re: WTF? W-here T-o F-ind the original hpmuseum forum?](#), Garth Wilson - 9 Dec 2013, 8:53 p.m. **1 response**

[HP Prime finally updated!](#), Les Koller - 9 Dec 2013, 8:08 p.m. **8 responses**

[makemat on HP Prime](#), Jeff Nye - 9 Dec 2013, 7:04 p.m. **9 responses**

[OT--Nat'l Semiconductor NS-7100 pics anywhere?](#), Matt Agajanian - 9 Dec 2013, 6:44 p.m. **25 responses**

[HP Prime Matrices curiosity](#), bluesun08 - 9 Dec 2013, 6:44 p.m. **0 responses**

[HP Prime CAS curiosity](#), bluesun08 - 9 Dec 2013, 5:48 p.m. **11 responses**

[Silly me... I also bought a 12C and 15C](#), Eelco Rouw - 9 Dec 2013, 2:53 p.m. **3 responses**

[Need help understanding math....](#), cyrille de Brébisson - 9 Dec 2013, 1:40 p.m. **9 responses**

[Re: The Lucky WP34S Cable](#), eri - 9 Dec 2013, 1:40 p.m. **4 responses**

[Is Linux common among us RPN types?](#), db (martinez, ca.) - 9 Dec 2013, 1:26 p.m. **46 responses**

[WP-34S \(Emulator Program Load/Save\)](#), Barry Mead - 9 Dec 2013, 12:50 p.m. **1 response**

[HP 50g : More on best rechargeable batteries](#), Miguel Toro - 9 Dec 2013, 12:38 p.m. **1 response**

[Google honors Grace Hopper today](#), Don Shepherd - 9 Dec 2013, 10:28 a.m. **19 responses**

[hp-prime solver and variable name](#), fabrice48 - 9 Dec 2013, 4:47 a.m. **22 responses**

[Sending little images to the Prime \(...and Program name bug?\)](#), Erwin Ried - 9 Dec 2013, 3:06 a.m. **19 responses**

[hp-67 card reader problem.](#), Marcel Samek - 9 Dec 2013, 2:36 a.m. **5 responses**

[rolynchupetin hp50](#), lbintz4 - 9 Dec 2013, 12:48 a.m. **3 responses**

[HP35s Calculator Max Rope Tension Program](#), Jean-Marc Biram (Australia) - 8 Dec 2013, 9:40 p.m. **10 responses**

[\[WP-34S\] Unfortunate key damage with update to V3 :\(](#), svisvanatha - 8 Dec 2013, 8:35 p.m. **5 responses**

[HP-48GX & 50G Question](#), Matt Agajanian - 8 Dec 2013, 7:59 p.m. **2 responses**

[Usefulness of Woodstock Cs without their C](#), Matt Agajanian - 8 Dec 2013, 6:39 p.m. **5 responses**

[Re: Use 82240B as computer printer](#), R. Pienne - 8 Dec 2013, 6:31 p.m. **1 response**

[Re: A prime failure](#), Jan - 8 Dec 2013, 5:16 p.m. **0 responses**

[HP67 eBay auction](#), Cristian Arezzini - 8 Dec 2013, 3:24 p.m. **3 responses**

[Re: Updating Prime](#), aurelio - 8 Dec 2013, 2:44 p.m. **0 responses**

[Re: HP Prime Tutorial #6 and #7: Drawing](#), Giancarlo - 8 Dec 2013, 1:42 p.m. **1 response**

[Re: WTF? W-here T-o F-ind the original hpmuseum forum?](#), DeboT - 8 Dec 2013, 1:10 p.m. **2 responses**

[Re: Updating Prime](#), JavierLopez - 8 Dec 2013, 12:10 p.m. **0 responses**

[Re: F1227A as HP48 case: yes!](#), Harold A Climer - 8 Dec 2013, 11:08 a.m. **0 responses**

[Re: F1227A as HP48 case: yes!](#), megarat - 8 Dec 2013, 7:49 a.m. **2 responses**

[Re: HP Prime: Quite Happy](#), Joe Horn - 8 Dec 2013, 7:26 a.m. **1 response**

[Re: Bought a 16C to compensate a little](#), Nick_S - 8 Dec 2013, 6:02 a.m. **2 responses**

[Re: Improvements for productivity for the Prime](#), Stefan Dröge (Germany) - 8 Dec 2013, 5:11 a.m. **0 responses**

responses

Prime still frozen, Les Koller - *8 Dec 2013, 4:01 a.m.* **4 responses**

Re: Bought a 16C to compensate a little, Didier Lachieze - *8 Dec 2013, 2:20 a.m.* **1 response**

Re: WTF? W-here T-o F-ind the original hpmuseum forum?, Walter B - *8 Dec 2013, 1:47 a.m.* **8 responses**

OT: 3D Calculator Remakes, hugh steers - *28 Jan 2013, 8:54 a.m.* **0 responses**

Re: Sourceforge, Namir - *2 Feb 2012, 12:51 a.m.* **0 responses**

Odd looking 41cx, Chris Woodhouse - *10 Jan 2012, 12:31 p.m.* **0 responses**

Re: HP-15C - useless today?, Dave Mabry - *11 Dec 2011, 7:45 p.m.* **0 responses**

HP-15c LE Bad Keys, Gerry Schultz - *23 Sept 2011, 12:54 p.m.* **0 responses**

Re: 97 printer paper not advancing, Keith Beyer - *5 July 2010, 5:54 p.m.* **0 responses**

[[Top of Index](#) | [Daily view](#)]



[Go to the HP Forum archives selection page](#)



[Go back to the main exhibit hall](#)



OLD HP Forum (daily view)

[[New Topic](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
[[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) |
[Ads without Auctions](#)]

Welcome back!

Your last visit began on 30 Oct 2014, 10:34 p.m.

Since then, no new messages have been posted.

This is the OLD HP forum. For newer discussions, please visit the [New HP Forums](#).

How to register at the new forum:

Even if you had an account on the old forum, you'll need to create one for the new forum. In the upper left near "Hello There, Guest" you should see a register link.

Using the same name (or close) as as you used on the old forum is preferred. The new forum does not allow commas in names. If you had a comma before, please omit it from your new account name.

Message Index

Daily view for frequent visitors.

Messages Posted or Modified Within the Last Two Days

(Compressed "Threads Only" Listing)

No messages matched your search criteria! Please try again....

[[Top of Index](#) | [Daily view](#)]



[Go to the HP Forum archives selection page](#)

[Go back to the main exhibit hall](#)



HP Forum Archives

- [Forum Archive 21](#): March 2012 -
- [Forum Archive 20](#): January 2011 - March 2012
- [Forum Archive 19](#): February 2009 - January 2011
- [Forum Archive 18](#): November 2007 - February 2009
- [Forum Archive 17](#): March 2007 - November 2007
- [Forum Archive 16](#): March 2006 - March 2007
- [Forum Archive 15](#): April 2005 - March 2006
- [Forum Archive 14](#): Dec 2003 - April 2005
- [Forum Archive 13](#): May 2003 - Dec 2003
- [Forum Archive 12](#): April 2003 - May 2003
- [Forum Archive 11](#): February 2003 - April 2003
- [Forum Archive 10](#): January 2003 - February 2003
- [Forum Archive 9](#): September 2002 - January 2003
- [Forum Archive 8](#): April 2002 - September 2002
- [Forum Archive 7](#): December 2001 - April 2002
- [Forum Archive 6](#): May - December 2001
- [Forum Archive 5](#): February 2001 - May 2001
- [Forum Archive 4](#): October 2000 - February 2001
- [Forum Archive 3](#): July 2000 - October 2000
- [Forum Archive 2](#): December 1999 - July 2000
- [Forum Archive 1](#): March 1999 - December 1999



[Go to the current HP forum](#)



[Go back to the main exhibit hall](#)



HP Memories Forum

[[New Memory](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
[[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) |
[Ads without Auctions](#)]

Welcome back!

Your last visit began on 30 Oct 2014, 12:55 p.m.

Since then, no new memories have been posted.

This forum is intended to post memories related to HP calculators. Please **don't** post questions or advertisements here - there are other places on this web site for those and this forum does not allow replies. For comprehensive articles, see the Articles Forum. If you want to refer to one of these articles in the general forum, please refer to it by name or article number

Please use the [HP Calculator Ads](#) section for "for sale" and "wanted" ads. Please review the [Terms of Use](#) for more complete information.

It's possible to use some [advanced formatting techniques](#) in this forum if you wish.

Memory Index

Memories Posted or Modified Within the Last Archive

*175 of 175 Memories Displayed
(Reversed Threaded Listing)*

- 559: HP34C again!!!**, Stephan Matthys - 21 Aug 2013, 1:52 p.m.
- 556: Working up to the HP71B**, Nick R - 12 Dec 2010, 6:56 p.m.
- 554: The history of my calculators**, SteveH - 15 Oct 2010, 8:57 a.m.
- 553: Enter > Equal**, Steven F Johnson - 11 Feb 2010, 7:34 p.m.
- 549: HP-35: an Epiphany**, Doug Winkeljohn - 26 Jan 2010, 2:58 p.m.
- 547: HP-67 saves my holidays**, Klaus - 8 Oct 2009, 4:49 a.m.
- 545: HP-25 to the rescue and now, rescued!**, Joe Buczek - 28 Aug 2009, 10:57 p.m.
- 541: My personal history with calculators**, hecube - 9 Aug 2009, 1:12 a.m.
- 540: 36 years of HP ownership**, Philip Wakeling - 1 July 2009, 4:00 p.m.
- 539: a very strange hp-35**, Dana Bingham - 29 June 2009, 4:51 p.m.
- 538: i invite you all to look at message #69**, Dana Bingham - 23 June 2009, 2:29 p.m.
- 537: Ahh... HP calculators**, bbeamer - 10 June 2009, 11:59 a.m.
- 536: those were the days ...**, Glenn Becker - 29 May 2009, 8:54 a.m.
- 535: My quest for the elusive HP 42S**, Jeff Kearns - 18 Dec 2008, 10:24 p.m.
- 533: 1981 to 2008 my HP journey**, Ed Austin - 20 Oct 2008, 11:06 p.m.
- 531: What a shock .. what a relief**, peter a walker (papwalker) - 18 Aug 2008, 11:58 a.m.
- 526: HP-97 My First Computer**, MikeO - 21 June 2008, 6:50 p.m.
- 525: HP41-Article on german famous online-newsletter**, Andre Koppel - 14 May 2008, 10:39 a.m.
- 523: HP DESIGNS**, designnut - 24 Apr 2008, 9:03 p.m.
- 522: Let's call it... HP and I**, Alejandro Paz (Germany) - 12 Apr 2008, 5:46 p.m.
- 519: My HPs and me**, Kareem Mokdad - 3 Feb 2008, 2:10 p.m.
- 518: How I got started with HP calculators**, Martin Pinckney - 30 Aug 2007, 1:32 p.m.

- 517: [HP 2000 Access TS Basic](#), Davis S. MOORE - 24 Aug 2007, 12:58 a.m.
- 501: [Still Have My Faithful HP-15C](#), Candy Humphries - 14 July 2007, 6:05 p.m.
- 492: [My HP41CV Passed Away Today](#), Karin Seritis - 5 Apr 2007, 7:12 p.m.
- 491: [HP35](#), Keith Taylor - 3 Apr 2007, 2:06 p.m.
- 490: [HP Memory Project](#), Marc Mislange - 17 Mar 2007, 5:44 a.m.
- 487: [The Perfect EE Undergrad Calculator: HP-15c](#), Steven Staton - 17 Feb 2007, 2:02 a.m.
- 484: [HP, a Dream Come True and More](#), BartD - 21 Dec 2006, 1:02 p.m.
- 483: [HP41C.COM \(HP41 in Space\)](#), Ed Langley - 19 Dec 2006, 10:24 p.m.
- 479: [HP-45 calculator](#), Harry A. Madden - 20 Nov 2006, 1:15 p.m.
- 477: [Discovered the HP-41C](#), Mark Meier - 26 Oct 2006, 11:26 p.m.
- 476: [35 to 15C](#), michael thornton - 26 Oct 2006, 7:10 a.m.
- 375: [converted](#), e.young - 20 July 2006, 5:20 p.m.
- 296: [My first HP experience](#), Antonio Maschio (Italy) - 15 May 2006, 11:47 a.m.
- 284: [My Story From the 70's](#), John Lovda - 11 Apr 2006, 2:34 p.m.
- 279: [HP's I've owned](#), Voyager85 - 29 Mar 2006, 10:09 p.m.
- 277: [HP 12C ^ 2](#), C. F. Howlett - 16 Jan 2006, 7:33 p.m.
- 275: [Woodstocks rule](#), Hal - 15 Jan 2006, 3:49 a.m.
- 274: [HP28C](#), Henrik Carlsen - 13 Jan 2006, 3:28 a.m.
- 272: [Memories from a young 'un](#), Han Duong - 11 Nov 2005, 9:28 p.m.
- 270: [From Sinclair Scientific to HP-45 - What a Relief!](#), Les Bell - 22 Oct 2005, 3:20 a.m.
- 264: [HP Calculator Users Were the Most Passionate Customers](#), Dale Zezula - 25 Sept 2005, 9:15 p.m.
- 262: [The calc that always gave PI as the result](#), Marcus von Cube, Germany - 9 Aug 2005, 11:07 a.m.
- 261: [33 years with HP calculators](#), Mark Crispin - 7 Aug 2005, 3:06 p.m.
- 260: [All we have are memories](#), Gary D. Snyder (not the poet) - 3 Aug 2005, 7:32 p.m.
- 258: [11C aspect ratio](#), Dave Chapman - 27 July 2005, 1:31 p.m.
- 256: [Oh, you're not a freshman!](#), David Krider - 11 July 2005, 9:52 p.m.
- 252: [First look at an HP-35](#), Steve Talley - 6 May 2005, 2:21 p.m.
- 251: [HP narrative history](#), John Minck - 1 Apr 2005, 9:13 p.m.
- 247: [My HP-25 and being able to program!](#), Michael Brown - 16 Dec 2004, 6:00 p.m.
- 246: [11c](#), John - 21 Nov 2004, 3:13 a.m.
- 243: [Experience with the disintegrating HP-41C-system \(and batteries\).](#), Guido Abts - 20 Aug 2004, 4:13 a.m.
- 242: [Wow- what a tribute to this very cool calculator](#), Dale Gombert - 19 Aug 2004, 11:30 p.m.
- 241: [Someone Else's Loss](#), OJM - 12 July 2004, 12:21 p.m.
- 240: [HP-41CV WOW factor](#), Ted Pantenberg - 16 May 2004, 5:44 p.m.
- 239: [A successor to my slide rule](#), Aaron Chacon - 10 May 2004, 11:44 a.m.
- 229: [message archive](#), juanita boggs - 18 Apr 2004, 6:43 p.m.
- 228: [an ode to my calculator](#), Laura J - 11 Apr 2004, 5:07 p.m.
- 227: [Some fond memories of my CX](#), Fred Lennox - 24 Feb 2004, 9:05 p.m.
- 224: [HP-41. Beyond calculators. A real controller.](#), Garth Wilson - 19 Feb 2004, 10:15 p.m.
- 222: [the HP11C & the Sewage Pump Station](#), Wayne Stephens - 28 Jan 2004, 1:26 p.m.
- 221: [The HPs I had...](#), Jim Brown - 27 Jan 2004, 2:43 a.m.
- 219: [A simple truth for his Ignorant little sister:](#), Ron Ross - 17 Dec 2003, 3:46 p.m.
- 218: [Racy Machines](#), Palmer O. Hanson, Jr. - 11 Dec 2003, 3:15 a.m.
- 217: [...the old department store...](#), harryb - 12 Nov 2003, 10:46 a.m.
- 216: [HP-25 still running strong 28 years later...](#), Steve Eddy - 28 Oct 2003, 12:34 p.m.
- 215: [HPcalc memories](#), Wouter Peters - 21 Oct 2003, 2:56 p.m.
- 213: [First Exposure to Bill and Dave's World](#), Matt Kernal - 10 Oct 2003, 9:22 p.m.
- 200: [True Love Never Dies](#), Erich Neubacher - 2 Sept 2003, 5:05 p.m.
- 199: [An 11C story...](#), Regis Corbel - 30 July 2003, 9:19 a.m.
- 198: [33E Memories](#), Winston Smith - 24 July 2003, 9:10 a.m.
- 197: [A snowy day at a Colorado ski resort!](#), Bernie DiTullio - 10 July 2003, 5:01 p.m.
- 189: [HP-35 and the Soviets](#), Happy Holden - 1 July 2003, 5:54 p.m.
- 187: [HP Calculator Music: John Denver, Paul Simon](#), Norm - 28 June 2003, 1:43 a.m.

- [184: Nathan Pritikin's Diet and HP Calculators](#), Happy Holden - 22 May 2003, 12:20 p.m.
- [181: Working with Hewlett on the HP-35](#), Happy Holden - 9 Apr 2003, 2:18 p.m.
- [180: HP-34C BETTER THAN S*X](#), Norm - 30 Mar 2003, 3:18 a.m.
- [179: My HP story](#), Howard - 28 Mar 2003, 6:41 a.m.
- [178: HP-34C in high school calculus, back in 1980](#), Norm Hill - 27 Mar 2003, 12:06 a.m.
- [176: LEDs, TIs, and HPs.](#), Jeremy - 23 Mar 2003, 9:46 p.m.
- [174: HP48 and dialog boxes](#), Karma Policer - 23 Feb 2003, 11:51 p.m.
- [173: Buyer's Remorse... no more.](#), Michael Meyer - 29 Jan 2003, 11:53 p.m.
- [172: Fun with HP-11c's](#), surfguy - 21 Dec 2002, 2:17 p.m.
- [171: HP 35/45 competing with a main frame computer](#), Jan (Switzerland) - 6 Dec 2002, 10:21 a.m.
- [161: Calculator or Pocket Computer?](#), HPunk (Brazil) - 23 Nov 2002, 3:04 p.m.
- [160: Eulogy for a 15C](#), deb - 15 Nov 2002, 2:24 p.m.
- [155: 30 years of HP calculators](#), Tom Mathews - 12 Nov 2002, 1:04 p.m.
- [152: Persistence pays off!](#), Grant Henson - 4 Nov 2002, 11:30 a.m.
- [151: My First HP-21](#), Olivier TREGER - 1 Oct 2002, 8:27 a.m.
- [149: Way back in 1981...](#), EdAustin - 21 Sept 2002, 10:37 a.m.
- [148: hp80](#), sergio ricardo moreno - 4 Sept 2002, 4:28 a.m.
- [147: "Playing" with a 9820](#), Joe Hallett - 24 Aug 2002, 3:53 p.m.
- [146: HP Computing in 1969/1970](#), Chuck Lytle - 21 Aug 2002, 10:28 a.m.
- [145: my first paid program...](#), Clif Swinford - 20 Aug 2002, 9:40 p.m.
- [144: OOPs ! Wrong Battery in a HP-41CV](#), Eduardo Munoz - 31 July 2002, 8:40 p.m.
- [142: Another memory from me: HP48G.](#), HPhreaker - 23 June 2002, 10:06 p.m.
- [140: HP memories from Brazil.](#), HPhreaker - 2 June 2002, 5:45 p.m.
- [137: HP48GX/HP49G](#), George Tsiros - 26 May 2002, 10:28 a.m.
- [135: HP-42S](#), Walter Christensen - 10 May 2002, 3:45 p.m.
- [134: My first calculator](#), David Chong - 23 Apr 2002, 1:20 a.m.
- [133: MIT physics lab and HP-45](#), Paul M Lieberman - 22 Apr 2002, 5:07 p.m.
- [132: paper tape punch 9884A](#), Milan Jolic - 16 Apr 2002, 5:47 p.m.
- [131: HP 21 & 60 seconds of "++++..."](#), Robert Miller - 15 Apr 2002, 11:45 a.m.
- [127: The story of a globe trotting HP-15C](#), Ala'a H. Jawad - 1 Apr 2002, 7:45 a.m.
- [124: H-P Calc trivia fm ex employee](#), Jim Grant - 22 Feb 2002, 11:17 a.m.
- [117: memories? Nope, my 35 still works](#), Bob Hartman - 7 Feb 2002, 3:07 p.m.
- [116: HP Memories from Greece](#), Nick Karagiaourogrou - 23 Jan 2002, 3:22 p.m.
- [113: Floored and thunderstruck by the HP-35](#), D.Slauson - 21 Jan 2002, 5:17 p.m.
- [112: Funny anecdote about HP-45](#), aruid - 18 Jan 2002, 6:35 p.m.
- [108: The great trip East with the 41C](#), John Hancock - 7 Dec 2001, 1:07 a.m.
- [107: HP-35 vs Slide Rule](#), Bob Hermann - 28 Nov 2001, 3:51 p.m.
- [106: First time I was paid for programming](#), Nelson Asinowski - 16 Nov 2001, 3:45 p.m.
- [102: HP Memories](#), Fred Stone - 25 Oct 2001, 1:57 p.m.
- [97: Thanks for the magic memories](#), Carl Landsness - 13 July 2001, 9:55 p.m.
- [96: Truck ran over my HP-11C](#), Tim Souza - 7 July 2001, 1:44 a.m.
- [95: Your 41 did what?](#), Jim Creybohm - 23 June 2001, 11:22 p.m.
- [92: HP41C Memories](#), Namir Shammass - 20 June 2001, 2:21 p.m.
- [91: My first HP-35 and several HP's later](#), C_Thomas - 12 June 2001, 8:11 p.m.
- [90: My calculator is older than I am?](#), Neil Fraser - 8 June 2001, 10:48 p.m.
- [89: How I got my 65 before they were introduced](#), Dean Lampman - 30 Apr 2001, 6:18 p.m.
- [83: HP Memories and my HP-80](#), Richard Baumann - 12 Apr 2001, 8:32 a.m.
- [78: Happy to see the HP calculator site.](#), Stan Shankman - 12 Apr 2001, 1:15 a.m.
- [77: my addiction](#), Ron Ross - 28 Mar 2001, 5:25 p.m.
- [76: Only one calculator for me](#), Ernie Malaga - 13 Mar 2001, 2:34 p.m.
- [74: The Magic of Continuous Memory](#), Paul Brogger - 20 Feb 2001, 3:19 p.m.
- [73: Once More Through The Rinse Cycle, Please!](#), Brian W. Haren - 16 Feb 2001, 10:28 a.m.
- [72: Landing on the moon never impressed me](#), D. Banks - 12 Feb 2001, 10:25 a.m.
- [69: Best money i ever spent](#), Dana Bingham - 6 Feb 2001, 11:29 p.m.

- 68: [HP41c Memories](#), Mike O'Regan - 29 Jan 2001, 6:18 p.m.
- 67: [Treasure your classic HPs](#), Fred Boness - 19 Jan 2001, 3:07 a.m.
- 66: ["Funny, for that much, you'd think it would be bigger."](#), Rick Wagoner - 15 Jan 2001, 3:30 p.m.
- 64: [Nostalgia](#), Len Matlaw - 31 Dec 2000, 1:05 p.m.
- 62: [My HP 67](#), Rex Wiederanders - 7 Nov 2000, 10:28 p.m.
- 61: [HP-35 and beyond](#), Erik Wahlin - 16 Sept 2000, 12:58 a.m.
- 60: [HP 9100 Environmental Controller and Telemetry Systems](#), Wirt Atmar - 10 Sept 2000, 1:03 p.m.
- 59: ["new" memory](#), john - 5 Sept 2000, 6:46 p.m.
- 56: [16-yr old HP-15C still running on original batteries](#), Chuck Yount - 1 Sept 2000, 12:24 p.m.
- 55: [Some 41 memories](#), Juan Jijon - 26 Aug 2000, 7:39 p.m.
- 53: [HP-35 Unfair Advantage \(plus HP-67 story\)](#), John Meyer - 24 Aug 2000, 2:21 p.m.
- 52: [an interesting hp device, etc.](#), Robert Weigel - 11 Aug 2000, 1:39 a.m.
- 51: [My HP-3545](#), Andrew Burg - 28 July 2000, 12:45 a.m.
- 48: [HP-41 fond memories](#), Chris Weaver - 27 June 2000, 7:29 p.m.
- 47: [Good ol' 28C!](#), Matt J. McCullar - 25 June 2000, 2:31 p.m.
- 45: [Indestructible 11C](#), Maurice Delaney - 12 June 2000, 9:33 a.m.
- 43: [HP35](#), Ian Kettleborough - 22 May 2000, 3:27 a.m.
- 41: [A Godsend HP42S](#), John Triplett - 18 May 2000, 3:16 p.m.
- 34: [On the Eve of My 2nd Birthday](#), Bill Duncan, ve3ied - 22 Apr 2000, 9:25 p.m.
- 33: [HP-16 RIP](#), Steve - 22 Apr 2000, 10:39 a.m.
- 28: [Reactivating an old HP87](#), Fred Ellers - 8 Apr 2000, 3:55 p.m.
- 26: [HP-25 Controlled Coffee Maker](#), francis TEYSSIER - 2 Apr 2000, 10:27 p.m.
- 25: [tticon@swbell.net](#), Barry Williams - 2 Apr 2000, 10:26 p.m.
- 24: [Where is the Equal sign?](#), E. Abgarian - 2 Apr 2000, 10:24 p.m.
- 23: [Car Trek XVI -- The Search for the HP-16C](#), Wayne Brown - 2 Apr 2000, 10:22 p.m.
- 22: [Calculator Memories from Argentina](#), Andrés C. Rodríguez - 2 Apr 2000, 10:17 p.m.
- 21: [Collecting Memories](#), Dr. Sean F. Johnston - 2 Apr 2000, 10:16 p.m.
- 20: [You Could Have Bought A Good Used Car For That!](#), Mike Morrow - 2 Apr 2000, 10:15 p.m.
- 19: [It Still Works As Good As The Day I Bought It](#), Steve Simpkin - 2 Apr 2000, 10:14 p.m.
- 18: [A Miniature 9100](#), Adams Douglas - 2 Apr 2000, 10:13 p.m.
- 17: [Nobody Wanted To Touch It](#), Bruce Kebbekus - 2 Apr 2000, 10:12 p.m.
- 16: [HP41C: More Memories](#), Raul F. Rueda - 2 Apr 2000, 10:11 p.m.
- 15: [HP-41 From Inside](#), Russell Nelson - 2 Apr 2000, 10:10 p.m.
- 14: [Starting Your Kids On The Right Path](#), Chris Lott - 2 Apr 2000, 10:09 p.m.
- 13: [My Calulatin' life](#), Tom Lianza - 2 Apr 2000, 10:08 p.m.
- 12: [HP-35 "Know-it-all"](#), Mik Butler - 2 Apr 2000, 10:06 p.m.
- 11: [JPL Meets the HP-35](#), Dan Bathker - 2 Apr 2000, 10:04 p.m.
- 10: [An HP made a hacker out of me.....](#), Alexandra L. Carter - 2 Apr 2000, 9:59 p.m.
- 9: [HP-25 Survives Montana Winter](#), Julius Strid - 2 Apr 2000, 9:58 p.m.
- 8: [HP-41 Memories](#), Dan Grelinger - 2 Apr 2000, 9:57 p.m.
- 7: [HP Mystique](#), Terry Nielsen - 2 Apr 2000, 9:56 p.m.
- 5: [My experiences in collecting HP and how I came to it](#), Thomas Kiesewetter - 2 Apr 2000, 9:55 p.m.
- 4: [HP-25 Reminiscence and Regret](#), Richard Woods - 2 Apr 2000, 9:54 p.m.
- 3: [Thanks For The Memories](#), John Alwood - 2 Apr 2000, 9:52 p.m.
- 2: [Constructive Butchery of a HP25C](#), Walter Gray - 2 Apr 2000, 9:51 p.m.
- 1: [Unshakable Bond with \(almost\) Unbreakable HP](#), Paul J. Brogger - 2 Apr 2000, 9:49 p.m.

[[Return to the Message Index](#)]



[Go to the HP Forum archives selection page](#)



[Go back to the main exhibit hall](#)



HP Biographies

[[New Biography](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
 [[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) | [Ads without Auctions](#)]

Welcome back!

Your last visit began on 30 Oct 2014, 12:55 p.m.
 Since then, no new biographies have been posted.

This is a place where you can enter your biography and contact data. Feel free to put as much or as little information here as you like.

Biography Index

Biographies Posted or Modified Within the Last Archive

[Abraham, Ion](#) (New Mexico, USA) 17 Oct 2002, 8:49 p.m.
[Alan Taylor](#) (Farnborough Aerospace Centre UK) 18 Mar 2007, 9:04 p.m.
[Alberto Fenini](#) (Milano, Italy) 6 Sept 2011, 5:10 a.m.
[Austin, Edward](#) (Russia - UK) 1 Nov 2008, 8:43 p.m.
[Becker, Glenn E.](#) (Arlington, MA) 30 May 2009, 1:54 p.m.
[Bell, Les](#) (Sydney, Australia) 22 Oct 2005, 2:07 a.m.
[Bensene, Rick](#) (Oregon, USA) 1 Oct 2002, 8:34 p.m.
[Bradford, George](#) (Atlanta, GA (USA)) 17 July 2007, 11:01 a.m.
[Brian Haren](#) (Atlanta, Georgia) 10 June 2007, 8:27 a.m.
[Butts, Ed](#) (Salem Oregon) 17 July 2005, 3:49 p.m.
[Chrapkiewicz, Thomas](#) (Detroit, Michigan, USA) 21 Aug 2007, 3:07 p.m.
[Clark, Keith](#) (Floral city Florida) 8 Sept 2003, 7:57 p.m.
[Corbel, Régis](#) (Lannion, Brittany (western part of France)) 5 Aug 2003, 9:37 a.m.
[Corrado, Larry](#) (Wisconsin, USA) 21 Feb 2003, 9:41 a.m.
[Crawford, Peter D.](#) (Chicago) 11 Dec 2008, 4:56 p.m.
[Duell, Tony](#) (London, England) 12 Apr 2004, 9:48 a.m.
[Easterling, Stephen](#) (Melbourne, FL) 6 Dec 2005, 9:28 p.m.
[Frenkel, Mallory Patrick](#) (Karlsruhe, Germany) 21 Jan 2003, 4:11 a.m.
[Fubar Vikinghelmet](#) (New York City) 8 Feb 2005, 12:38 p.m.
[Gnerucci, Massimo](#) (Milano - Italy) 30 Mar 2002, 2:27 a.m.
[Haley, Scott](#) (Broken Arrow, OK) 8 Dec 2009, 11:25 a.m.
[Halvorsen, Arne](#) (Norway, Bergen) 23 Aug 2007, 3:56 p.m.
[Hamilton, Neil](#) (Ottawa, Canada) 19 May 2011, 7:23 p.m.
[Hancock, John](#) (Adelaide, Ausrtalia) 7 Dec 2001, 12:41 a.m.
[Harper, Christof](#) (Davis, CA USA) 15 Sept 2002, 1:42 p.m.
[Hayden, David](#) (Crosswicks, New Jersey, USA) 21 Feb 2010, 7:57 a.m.
[Hicks, David G.](#) (USA) 21 Oct 2001, 7:44 p.m.
[Hicks, Mike](#) (Olathe, KS) 19 Jan 2002, 4:18 p.m.
[Holden, Happy](#) (Loveland, CO) 27 May 2003, 4:33 p.m.
[Horn, Jim \(James L.\)](#) (Hood River, Oregon USA) 21 Sept 2010, 6:36 p.m.
[Hyde, Bill](#) (Santa Fe, New Mexico -- USA) 2 Jan 2008, 3:25 p.m.
[Jamaludin, Arjunaidi](#) (Malaysia (MY, Earth)) 14 June 2008, 8:35 a.m.

Jorge Ibrahim Rodriguez (Pasadena, California) 8 June 2009, 4:21 a.m.
Keith Midson (Tasmania, Australia) 3 Oct 2012, 10:17 a.m.
Kercheval, John 30 Jan 2002, 8:46 p.m.
Kinoshita, Masao (Boston Area) 16 Sept 2002, 5:23 p.m.
Kinoshita, Masao (Boston area, USA) 13 Nov 2007, 1:05 p.m.
Levy, E Samuel (San Diego) 12 Apr 2007, 11:14 p.m.
Lilley, Robert (Belvidere, New Jersey USA) 6 Jan 2008, 9:40 a.m.
Lueck, Keith (St. Louis, Missouri, USA) 24 Oct 2007, 2:28 p.m.
Maguire, W. Bruce, II (Longmont, Colorado, USA) 26 Oct 2001, 3:13 p.m.
Marin, Paul (Melbourne, Australia) 19 Apr 2005, 7:19 p.m.
Mark Scheuern 23 June 2012, 9:37 a.m.
Mattioni, Giancarlo (Ancona - Italy) 6 July 2005, 4:14 p.m.
Merlino Chiozza, Sergio A. (Montevideo, Uruguay) 27 Aug 2012, 9:01 p.m.
Meyer, Michael (Omaha, NE) 12 Mar 2003, 10:10 p.m.
Mosand, John (Trondheim, Norway) 26 Oct 2001, 9:30 a.m.
Nelson, Vern (Winnipeg, Manitoba, Canada) 14 Nov 2005, 11:09 p.m.
Osman, Eric (Scholls, Oregon, USA) 16 Mar 2005, 12:38 p.m.
P., HPhreacker (Porto Alegre, Brazil) 13 Apr 2002, 6:49 p.m.
Paine, Cameron (Melbourne, Australia) 24 Oct 2001, 2:21 p.m.
Pio, Marx (Manaus, Amazonas, Brasil) 10 Nov 2001, 8:43 p.m.
Plant, Michael (Courtenay, BC, Canada) 14 Nov 2009, 7:27 p.m.
Platt, Bill (New England and the Mid Atlantic --a mass in motion) 8 July 2005, 11:04 p.m.
RAFE, Gary E. (Toledo, Ohio USA) 26 Mar 2003, 6:47 p.m.
Raia, Jerry (Los Angeles, CA USA) 15 Aug 2010, 11:05 p.m.
Randle, Chris (Lincoln, UK) 24 Oct 2001, 6:13 p.m.
Rasmussen, Johnny Bjoern (Denmark) 27 Apr 2004, 2:56 a.m.
Reed, Don 3 Dec 2013, 5:16 p.m.
Rincon, Gerardo (Michigan) 16 Apr 2010, 8:56 p.m.
Rodriguez, Osvaldo (El Paso, TX) 17 Aug 2007, 10:16 p.m.
Rodríguez, Andrés C. (Buenos Aires, Argentina) 5 Nov 2001, 6:37 p.m.
Ron Ross (Maryland/Delaware) 28 Mar 2003, 4:04 p.m.
Shaw, Kenneth (Toronto, Canada) 27 Nov 2007, 2:35 p.m.
Shek, Johnny (Hong Kong) 29 June 2008, 9:35 a.m.
Shore, Eddie (Azusa, CA) 23 Oct 2004, 6:07 p.m.
Smith, Bill (Bend, Oregon) 2 Apr 2007, 8:58 p.m.
Smith, Mark (Hunter Valley, Australia) 5 Nov 2004, 12:17 a.m.
Speck (Illinois, USA) 16 Sept 2002, 12:33 a.m.
Steve Simpkin (Southern California Desert) 14 Oct 2008, 8:22 p.m.
Treger, Olivier (Paris, France) 2 Jan 2007, 6:40 p.m.
Vieira, Luiz Cláudio (Brazil (Minas Gerais)) 19 Dec 2001, 6:06 a.m.
Vorkoetter, Stefan (Ontario, Canada) 13 Mar 2008, 10:40 a.m.
Vulic, Nenad (Split, Croatia) 9 Oct 2002, 5:28 p.m.
Wasserman, Katie (Rye Brook, NY) 4 Aug 2009, 12:44 a.m.
Watmough, Jonathan (Houston Texas) 14 June 2005, 11:42 p.m.
Wehrli, Matthias (Switzerland) 28 Oct 2001, 2:51 p.m.
Wong, Yiu Wah (Hong Kong, China, Asia) 29 Nov 2001, 4:49 a.m.
Zorns, Jeremy (Round Lake Beach, IL (A northern Chicago suburb)) 11 Mar 2003, 7:43 p.m.

[[Return to the Biography Index](#)]



[Go to the HP Forum archives selection page](#)

[Go back to the main exhibit hall](#)





HP Calculator Ads

[[New HP advertisement](#) | [Rules](#) | [Ad FAQ](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
 [[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) | [Ads without Auctions](#)]

Welcome back!

Your last visit began on 30 Oct 2014, 12:55 p.m.
 Since then, no new HP advertisements have been posted.

This forum is for the buying and selling of older (out-of-production) HP calculators, computers and parts and supplies as well as mechanical calculators and slide rules. Please do not publish ads for the same items or services more frequently than once per week. **Inappropriate or frequently reposted ads may be deleted at any time with or without notification.** Please review the [Terms of Use](#) for more complete information.

The Museum of HP Calculators and the curator take no responsibility for ads placed here. Please take appropriate precautions when dealing with strangers over the internet. **Please Beware of Scams!!** Please use the [HP forum](#) section for general discussions.

The following abbreviations are used:

(WTB) Want To Buy	(AUCTION) For sale by online Auction (like eBay)
(FS) For Sale at fixed price	(TRADE) For Trade (only)
(FSBO) For Sale at Best Offer	(FSOT) For Sale Or Trade
(MFSBO) May be For Sale at Best Offer	(SW) Service Wanted
(FSOBO) For Sale at fixed price Or Best Offer	(SO) Service Offered
(FREE) Free item(s)	

HP advertisement Index

The most recent ads are shown with previews. Click on a title to see a complete ad.

HP advertisements Posted or Modified Within the Last Archive

*24 of 24 HP advertisements Displayed
 (Reversed Threaded Listing)*

[HP 18C / HP 28C / HP 28S / HP 32S II \(AUCTION\)](#)

The next calculators of my collection are for sale | HP 18C | http://www.ebay.de/itm/251398932837?ssPageName=STRK:MESELX:IT&_trksid=p3984.m1558.l2649 | HP 28C | http://www.ebay.de/itm/251398931612?ssPageName=STRK:MESELX:IT&_trksid=p3984.m1558.l2649 | HP 28S | http://www.ebay.de/itm/251398930228?ssPageName=STRK:MESELX:IT&_trksid=p3984.m1558.l2649 | HP 32S II | http://www.ebay.de/itm/251398928836?ssPageName=STRK:MESELX:IT&_trksid=p3984.m1558.l2649
 [[Contact Daniel A.](#) | [View complete ad](#)] Posted 14 Dec 2013

[HP 9820A \(FSBO\)](#)

I've for sale a HP 9820A with working printer and card reader. | Calculator is in nice condition fully working with 2 rom cartridges. | Item is located in the Netherlands, will be shipped worldwide. | I'm open for offers..
 [[Contact Rik Bos](#) | [View complete ad](#)] Posted 10 Dec 2013

[HP 16C for sale \(AUCTION\)](#)

Selling my HP16C on ebay, item 231111969624 | Auction and buy it now.

[[Contact Morten Nygaard Åsnes](#) | [View complete ad](#)] Posted 9 Dec 2013

[48sx Overlays \(P/N 82220a and 82210a\) \(WTB\)](#)

Looking for the generic 82220a overlay kit for the 48sx and the 41CV Emulator Card Overlay (82210a). Thanks!

[[Contact svisvanatha](#) | [View complete ad](#)] Posted 8 Dec 2013

[HP 85B wiht QIC40 modified tape drive \(AUCTION\)](#)

I've placed a HP 85B (32kB + Mass storage rom inside) on ebay. | Item 370957485834. | Also on ebay a lot of other HP calculator computer related stuff.

[[Contact Rik Bos](#) | [View complete ad](#)] Posted 4 Dec 2013

[HP20b/HP30b Programming Cable \(WTB\)](#)

I'm looking for an HP20b/HP30b programming cable. I'm hoping that someone has one they don't need anymore. Thanks.

[[Contact Stephen Rose](#) | [View complete ad](#)] Posted 3 Dec 2013

[Rare HP Xpander Prototype \(AUCTION\)](#)

Selling one of my Xpanders from my collection. See ebay item: 231108514788

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 3 Dec 2013

[HP 11C BOXED COMPLETE \(AUCTION\)](#)

Selling my boxed HP 11C on ebay - in excellent condition. See ebay item 331079302382

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 3 Dec 2013

[HP95LX F1001A Connectivity Pack \(AUCTION\)](#)

Selling my HP F1001A connectivity pack for HP95LX pocket computer (HP95LX not included). Fully complete with cable, floppy discs, user manual, etc. Ebay item 331076200410

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 28 Nov 2013

[HP41CX Opt001 Blanknut \(AUCTION\)](#)

Selling my rare Blanknut HP-41CX calculator on ebay. In perfect condition, fully functional with overlay. Item number: 231105410392

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 28 Nov 2013

[HP Portable Plus accessories \(WTB\)](#)

Recently I required an almost "naked" Portable Plus, and now I am looking for printed (original) documentation and software rom modules, particularly programming languages, or other accessories. | Also if you want to sell a complete Portable Plus system, please make an offer.

[[Contact Michael Fehlhammer](#) | [View complete ad](#)] Posted 27 Nov 2013

[2014 pocket computer calendar \(FS\)](#)

Following the success of my 2014 HP calendar, I have also made a 2014 pocket computer calendar. It showcases HP along with other brands such as Sharp, Casio, Elektronika, Panasonic, etc. You can view and/or purchase here:

<http://www.redbubble.com/people/kmidson/calendars/11094561-vintage-pocket-computers>

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 26 Nov 2013

[Sparcom Modules & Other HP Books hard to find \(AUCTION\)](#)

Just listed on ebay 48GX/SX Sparcom modules for Mathematics, Chemistry, Mechanical Engineering, Electrical Engineering and Physics along with a bunch of HP Application books and manuals. I will be adding numerous items next week, including HP Calculators, Manuals and Accessories. These auctions close Nov 30 and Dec 1. | Some items added this week include mint versions of HP21S, HP48GX and manuals, HP 71B, HP10B sealed, New in box HP 20S, HP 15C LE Sealed, HP19BII with burgundy leather case & manual, 8 book calculator collector set, misc. HP items. | Opening bids are set reasonable for ...

[[Contact Frank Knight](#) | [View complete ad](#)] Posted 23 Nov 2013

[HP Streamsmart 400 with two probes \(AUCTION\)](#)

Ends Saturday November 30. | Ebay item: 301025115344

[[Contact gene wright](#) | [View complete ad](#)] Posted 23 Nov 2013

[Bringslid BAS Calculus Mathematics 1 Expansion Card for HP48 \(FSBO\)](#)

Expansion card for HP48 series with manual. | BAS Calculus New Mathematics 1 VER 3.0 | Produced by Odd Bringslid ISV Spenningsgate 11 3601 Kongsberg Norway | From the table of contents: | Integration | Algebra | Functions | Linear Equations | 2D Curves | Series | Complex Numbers | Functions of Several Variables | Diffequations | Numerical Methods | Solved Problems | 172 pages spiral bound manual and expansion card.

[[Contact Chris Pem10](#) | [View complete ad](#)] Posted 20 Nov 2013

[Xpander Prototype \(FS\)](#)

I am selling one of my HP Xpander prototypes. It is in mint condition and Comes with power cord, stylus, hard cover, CD manual (copy, not original), and computer connection cord. It does not come with the original box or user manual sorry. I can send you photos if you like - I have some on my Flickr collection site

<http://www.flickr.com/photos/60529780@N02/7854887210/in/set-72157626354800664> | If it doesn't sell here within a week or so then I will list on ebay. Asking US\$350 which includes postage to anywhere in the world (i'm located in Australia). Payment by PayPal or wire transfer. | T. . .

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 17 Nov 2013

[Sparcom Spice48 expansion card for HP 48 \(FSOT\)](#)

Hello, | I have a Sparcom/DaVinci Spice48 card that is looking for a new home (alas, without manual). Ideally I would do a trade, as I'm looking for either a Sparcom General Chemistry Reference Pac (with manual), or the original Educalc "leather desk" case for the HP 48-series calculators, or perhaps a Math Pro Pac. That said, if I can't find an appropriate trade, I'll be happy to sell it. | Please message me if you're interested.

[[Contact megarat](#) | [View complete ad](#)] Posted 13 Nov 2013

[2014 HP calendar \(FS\)](#)

Hi all, i've made a 2014 calendar from various photos from my collection. | It can be viewed and purchased from Redbubble - <http://www.redbubble.com/people/kmidson/calendars/11094118-hewlett-packard-calendar> | Hope you enjoy - it was hard to decide what photos were 'in' and 'out' with only 12 months. I may do another that includes vintage pocket computers. Let me know if there are any that you would like to see included. Cheers, Keith

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 13 Nov 2013

[HP-41 Flex-PCB replacement \(FS\)](#)

Hi all, | A new batch of Flex-PCB replacements is available at 22Euro/unit. | This time I've placed a 'Buy It Now' ad at eBay; item# 111210794951. | Cheers from Caribbean. | Diego Diaz. | (www.clonix41.org)

[[Contact Diego Diaz](#) | [View complete ad](#)] Posted 8 Nov 2013

[HP 10C + explanatory booklet for HP 10 series \(AUCTION\)](#)

Selling my HP 10C with some signs of usage and scratches on the backside. | http://www.ebay.de/itm/251372150942?ssPageName=STRK:MESELX:IT&_trksid=p3984.m1558.l2649 | And I'm also selling an explanatory booklet for HP 10 series calculators in German. | http://www.ebay.de/itm/251372155199?ssPageName=STRK:MESELX:IT&_trksid=p3984.m1558.l2649 | More detailed information inside the auctions.

[[Contact Daniel A.](#) | [View complete ad](#)] Posted 5 Nov 2013

[Sharp PC-1280 \(WTB\)](#)

Mint or new preferred. Please make an offer with pictures and full description.

[[Contact From Hong Kong](#) | [View complete ad](#)] Posted 1 Nov 2013

[Selling several calculator from HP 48/49 series \(AUCTION\)](#)

I want to sell my whole collection of HP calculators and started with some HP 48 and HP 49. Maybe there's something interesting for you in it. All of them are in good condition but see the auction details for more information. | HP 33 S | http://www.ebay.de/itm/HP-33-S-RPN-wissenschaftlicher-Taschenrechner-scientific-calculator-/251366671989?pt=DE_Technik_B%C3%BCrotechnik_Taschen_Tischrechner&hash=item3a869f0675 | HP 48 gII |

http://www.ebay.de/itm/HP-48-gII-RPN-wissenschaftlicher-Taschenrechner-scientific-calculator-/251366674559?pt=DE_Technik_B%C3%BCrotechnik_Taschen_Tischrechner...

[[Contact Daniel A.](#) | [View complete ad](#)] Posted 28 Oct 2013

[HP50g mint condition UK \(FS\)](#)

HP50G for sale due to recent underuse. | Comes with USB cable, 1Gb SD card, fresh batteries and soft case. Perfect condition. Current OS release. | Will accept £45 for collection from London (TW7 postcode) or £50 for delivery anywhere in the UK. | Available with two full sets of awesome Sanyo Eneloop rechargeable low self-discharge AAA batteries and a slow charger for an additional £15.

[[Contact Chris Smith](#) | [View complete ad](#)] Posted 25 Oct 2013

[HP 9807a ROM wanted to buy or loan \(WTB\)](#)

My HP integral powers up but does not boot. Can anyone sell or loan me a HP-UX ROM, BASIC or service ROM, so that I can test. Thanks.

[[Contact dbintley](#) | [View complete ad](#)] Posted 22 Oct 2013

[[Return to the full Index](#) | [Return to the Index without auctions](#)]



[Go to the HP Forum archives selection page](#)



[Go back to the main exhibit hall](#)



HP Calculator Ads (Without Auctions)

[[New HP advertisement](#) | [Rules](#) | [Ad FAQ](#) | [Search / Personalize](#) | [Mark All Read](#) | [Login / Register](#)]
[[HP Forum](#) | [Forum Daily View](#) | [Archives](#) | [Articles](#) | [Memories](#) | [Biographies](#) | [Classified Ads](#) | [Ads without Auctions](#)]

Welcome back!

Your last visit began on 30 Oct 2014, 12:55 p.m.
Since then, no new HP advertisements have been posted.

This forum is for the buying and selling of older (out-of-production) HP calculators, computers and parts and supplies as well as mechanical calculators and slide rules. Please do not publish ads for the same items or services more frequently than once per week. **Inappropriate or frequently reposted ads may be deleted at any time with or without notification.** Please review the [Terms of Use](#) for more complete information.

The Museum of HP Calculators and the curator take no responsibility for ads placed here. Please take appropriate precautions when dealing with strangers over the internet. **Please Beware of Scams!!** Please use the [HP forum](#) section for general discussions.

The following abbreviations are used:

(WTB)	Want To Buy	(AUCTION)	For sale by online Auction (like eBay)
(FS)	For Sale at fixed price	(TRADE)	For Trade (only)
(FSBO)	For Sale at Best Offer	(FSOT)	For Sale Or Trade
(MFSBO)	May be For Sale at Best Offer	(SW)	Service Wanted
(FSOBO)	For Sale at fixed price Or Best Offer	(SO)	Service Offered
(FREE)	Free item(s)		

HP advertisement Index (Without Auctions)

The most recent ads are shown with previews. Click on a title to see a complete ad.

HP advertisements Posted or Modified Within the Last Archive

*13 of 24 HP advertisements Displayed
(Reversed Threaded Listing)*

[HP 9820A \(FSBO\)](#)

I've for sale a HP 9820A with working printer and card reader. | Calculator is in nice condition fully working with 2 rom cartridges. | Item is located in the Netherlands, will be shipped worldwide. | I'm open for offers..

[[Contact Rik Bos](#) | [View complete ad](#)] Posted 10 Dec 2013

[48sx Overlays \(P/N 82220a and 82210a\) \(WTB\)](#)

Looking for the generic 82220a overlay kit for the 48sx and the 41CV Emulator Card Overlay (82210a). Thanks!

[[Contact svsvanatha](#) | [View complete ad](#)] Posted 8 Dec 2013

[HP20b/HP30b Programming Cable \(WTB\)](#)

I'm looking for an HP20b/HP30b programming cable. I'm hoping that someone has one they don't need anymore. Thanks.

[[Contact Stephen Rose](#) | [View complete ad](#)] Posted 3 Dec 2013

[HP Portable Plus accessories \(WTB\)](#)

Recently I required an almost "naked" Portable Plus, and now I am looking for printed (original) documentation and software rom modules, particularly programming languages, or other accessories. | Also if you want to sell a complete Portable Plus system, please make an offer.

[[Contact Michael Fehlhammer](#) | [View complete ad](#)] Posted 27 Nov 2013

[2014 pocket computer calendar \(FS\)](#)

Following the success of my 2014 HP calendar, I have also made a 2014 pocket computer calendar. It showcases HP along with other brands such as Sharp, Casio, Elektronika, Panasonic, etc. You can view and/or purchase here:

<http://www.redbubble.com/people/kmidson/calendars/11094561-vintage-pocket-computers>

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 26 Nov 2013

[Bringslid BAS Calculus Mathematics 1 Expansion Card for HP48 \(FSBO\)](#)

Expansion card for HP48 series with manual. | BAS Calculus New Mathematics 1 VER 3.0 | Produced by Odd Bringslid ISV Spenningsgate 11 3601 Kongsberg Norway | From the table of contents: | Integration | Algebra | Functions | Linear Equations | 2D Curves | Series | Complex Numbers | Functions of Several Variables | Diffequations | Numerical Methods | Solved Problems | 172 pages spiral bound manual and expansion card.

[[Contact Chris Pem10](#) | [View complete ad](#)] Posted 20 Nov 2013

[Xpander Prototype \(FS\)](#)

I am selling one of my HP Xpander prototypes. It is in mint condition and Comes with power cord, stylus, hard cover, CD manual (copy, not original), and computer connection cord. It does not come with the original box or user manual sorry. I can send you photos if you like - I have some on my Flickr collection site

<http://www.flickr.com/photos/60529780@N02/7854887210/in/set-72157626354800664> | If it doesn't sell here within a week or so then I will list on ebay. Asking US\$350 which includes postage to anywhere in the world (i'm located in Australia). Payment by PayPal or wire transfer. | T. . .

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 17 Nov 2013

[Sparcom Spice48 expansion card for HP 48 \(FSOT\)](#)

Hello, | I have a Sparcom/DaVinci Spice48 card that is looking for a new home (alas, without manual). Ideally I would do a trade, as I'm looking for either a Sparcom General Chemistry Reference Pac (with manual), or the original Educalc "leather desk" case for the HP 48-series calculators, or perhaps a Math Pro Pac. That said, if I can't find an appropriate trade, I'll be happy to sell it. | Please message me if you're interested.

[[Contact megarat](#) | [View complete ad](#)] Posted 13 Nov 2013

[2014 HP calendar \(FS\)](#)

Hi all, i've made a 2014 calendar from various photos from my collection. | It can be viewed and purchased from Redbubble - <http://www.redbubble.com/people/kmidson/calendars/11094118-hewlett-packard-calendar> | Hope you enjoy - it was hard to decide what photos were 'in' and 'out' with only 12 months. I may do another that includes vintage pocket computers. Let me know if there are any that you would like to see included. Cheers, Keith

[[Contact Keith Midson](#) | [View complete ad](#)] Posted 13 Nov 2013

[HP-41 Flex-PCB replacement \(FS\)](#)

Hi all, | A new batch of Flex-PCB replacements is available at 22Euro/unit. | This time I've placed a 'Buy It Now' ad at eBay; item# 111210794951. | Cheers from Caribbean. | Diego Diaz. | (www.clonix41.org)

[[Contact Diego Diaz](#) | [View complete ad](#)] Posted 8 Nov 2013

[Sharp PC-1280 \(WTB\)](#)

Mint or new preferred. Please make an offer with pictures and full description.

[[Contact From Hong Kong](#) | [View complete ad](#)] Posted 1 Nov 2013

[HP50g mint condition UK \(FS\)](#)

HP50G for sale due to recent underuse. | Comes with USB cable, 1Gb SD card, fresh batteries and soft case. Perfect condition. Current OS release. | Will accept £45 for collection from London (TW7 postcode) or £50 for delivery anywhere in the UK. | Available with two full sets of awesome Sanyo Eneloop rechargeable low self-discharge AAA

batteries and a slow charger for an additional £15.

[[Contact Chris Smith](#) | [View complete ad](#)] Posted 25 Oct 2013

[HP 9807a ROM wanted to buy or loan \(WTB\)](#)

My HP integral powers up but does not boot. Can anyone sell or loan me a HP-UX ROM, BASIC or service ROM, so that I can test. Thanks.

[[Contact dbintley](#) | [View complete ad](#)] Posted 22 Oct 2013

[[Return to the full Index](#) | [Return to the Index without auctions](#)]



[Go to the HP Forum archives selection page](#)



[Go back to the main exhibit hall](#)



The Museum of HP Calculators

Terms Of Use (Rules)

Last Updated June 5, 2002

Contents:

- [General Terms of Use](#)
- [Additional Terms of Use for Discussions About Ebay and Other Auction Sites](#)
- [Additional Terms of Use for the Classified Ads Section](#)

General Terms of Use

Please read this notice before using our site. By using the site, you agree to abide by these Terms of Use; please do not use our site if you do not agree with the Terms of Use outlined below. The Museum of HP Calculators (MoHPC) reserves the right to modify the Terms of Use at its sole discretion, and by using this site, you agree to be bound by the modifications or updates to the Terms of Use.

Users of this site will conduct themselves with responsibility and decorum, and will discuss topics of interest to our community. This should be a fun place to visit and all community members should feel welcome. The topics of these forums, include just about anything listed in the Museum pages. These include HP calculators, early HP computers, general math, calculating machines, computer history, tips on buying, finding, repairing, or using old HPs etc. Off-topic items may be deleted (or not) at the sole discretion of MoHPC.

You agree, through your use of this service, that you will not use this forum to post any material which is knowingly false and/or defamatory, inaccurate, abusive, belligerent, vulgar, hateful, harassing, obscene, profane, sexually oriented, threatening, or otherwise violative of any law.

You agree not to post any copyrighted material unless the copyright is owned by you or by this forum. You also agree not to use this forum to request or trade *copies* of copyrighted material. For example, most ROMs are copyrighted. Therefore, it is OK to buy and sell original ROM chips or calculators containing original ROM chips but in most cases it is *not* OK to buy and sell copies of those ROMs or ROM image files..

You agree to use a single consistent identity on the forums. MoHPC reserves the right to reveal the use of multiple identities, or to ban or moderate users using multiple identities, when it feels this is necessary. You are not required to use your real name but you must stick with one label.

Considering the real-time nature of this site, it is impossible for us to review messages or confirm the validity of information posted. Please remember that MoHPC does not actively monitor the contents of this board at all times and is not responsible for any messages posted. We do not vouch for or warrant the accuracy, completeness or usefulness of any message, and are not responsible for the contents of any message. The messages express the views of the author of the message, not necessarily the views of this forum or any entity associated with this forum. Any user who feels that a posted message is objectionable is encouraged to [contact us](#) immediately by email. We have the ability to remove objectionable messages and we will make every effort to do so, within a reasonable time frame, if we determine that removal is necessary. This is a manual process, however, so please realize that we may not be able to remove or edit particular messages immediately.

Although this forum does not and cannot review the messages posted and is not responsible for the content of any of

these messages, MoHPC reserves the right to delete, edit, or move any message for any reason or no reason whatsoever. As these Terms of Use are updated, old postings may or may not be retroactively deleted, edited, or moved at the sole discretion of MoHPC.

MoHPC reserves the right to prevent access to trolls, spammers, scammers and other disruptors, and to define which people meet those definitions. MoHPC may at times employ techniques that also prevent other users from accessing the forum via the same computer, LAN, or proxy.

You remain solely responsible for the content of your messages, and you agree to indemnify and hold harmless MoHPC and its agents with respect to any claim based upon transmission of your message(s).

The HP Forums are designed for members to express their opinions, viewpoints and testimonials on various products, services and events based upon speculation, personal knowledge and factual information through use, therefore all views represented here allow no liability upon the publishers of this web site and the owners of said views assume no liability for any ill will resulting from these postings. The posts made here are for educational as well as entertainment purposes and as such anyone viewing this portion of the website must accept these views as statements of the author of that opinion and agrees to release the authors from any and all liability.

You understand that information you place in a message will be available to anyone on the Internet for in indefinite period of time. If you use your email address, spammers may find it. (Accounts allow you to receive email without making your address public.) You and the museum moderators are not able to delete or alter posts that have made it to the archives, or that have been cached by search engines and spiders at other websites.

In addition to public forum messages, the forum software provides a mechanism that allows individuals to contact each other via email. The sender will only see your email address if you respond to the email. You understand and agree that these emails are coming from other various forum users with no involvement from MoHPC or forum moderators, and you agree to indemnify and hold harmless MoHPC and its agents with respect to any claim based upon transmission, reception and content of these messages. Copies of such email messages may or may not be retained by the web forum software and/or various servers and/or the recipient's email system.

If you do not wish others to be able to contact you via email, you can set your forum email address (NOT the activation address) to blank/empty and uncheck "Private email address".

The above rules will be strictly enforced. Continued violations may result in loss of HP Forum privileges and complaints being lodged with your Internet Service Provider.

Forum rules are not up for debate. This is not a democracy, but rather a private forum. You agree not to question the way MoHPC agents exercise their discretion in moderating and enforcing forum rules and guidelines.

Other than enforcing the above rules, the MoHPC does not attempt to control or validate any comments or recommendations provided by participants. As such, MoHPC is not responsible for any inaccuracies that may result in personal injury or damage to equipment.

HP Forum Commandments

1. Thou shalt remember that the vast majority of the posters on this forum are not employees or in any way related to MoHPC or Hewlett Packard, but are unpaid volunteers who donate hours of their time to provide information and assistance out of the goodness of their hearts, and without recompense.
2. Thou shalt not abuse, insult, harangue, blame, or take your anger out on same when you are experiencing problems, or we don't know the answer to your question.
3. Thou shalt remember that polite queries will elicit polite and helpful responses, likewise, that ranting and raving, bashing, and insults will prompt commensurate responses. Ye shall reap what ye sow.
4. Thou shalt remember that MoHPC provides this as a free service and its agents are not required to take bad behavior or insults by virtue of their position.
5. Thou shalt remember that we love to help out, so please don't be shy and feel free to ask any question regardless

of how silly you may feel asking it, and we'll be happy to help.

All rights reserved. By posting on this forum you grant The Museum of HP Calculators, and its successors and assigns a nonexclusive irrevocable right to re-use your posting in any manner it or they see fit without notice or compensation to you. No material contained in this site may be republished or reposted without express written consent of The Museum of HP Calculators, except of course that forum users retain the right to republish or repost their own work.

No warranty. This website and its services are provided "as is" and without any warranty or condition, express or implied.

In no event shall MoHPC or its owners be held liable for any damages whatsoever resulting from loss of use, loss of data, or loss of money.

Additional Terms of Use for Discussions About Ebay and Other Auction Sites

This is not meant to be a forum about ebay or other auction/sales sites though they will naturally come up at times. If you refer to any specific auction(s), sellers(s), or user(s) of any auction/sales site, then you must include your ID at that auction/sales site. This puts you on equal footing with the person you're talking about. Your auction/sales site ID may be confirmed by other forum users by sending a message via the auction/sales site. If that message isn't responded to in the affirmative, then your messages and/or account may be deleted. If you do not have an ID at the auction/sales site, or want to keep your identity secret, then please don't ask questions about specific auctions or sales on that site at this forum. If you have questions about a specific item, it is usually best to ask the seller for clarification. Also, please limit auction/sales threads to items of broad interest to HP calculator users.

Additional Terms of Use for the Classified Ads Section

The classified ads section is primarily intended to allow HP owners, users, and collectors to contact each other in a low pressure environment that is free of charges or fees. This implies certain differences from commercial dealer sites and auction sites. Dealers (or other higher volume sellers) are welcome but must take care not to offend others by over-advertising. If others are offended by repeated dealer advertisements, these ads may be deleted, edited and/or the seller may be asked to reduce the frequency of postings. Any user who feels that a posted message is objectionable, off-topic, or has been repeated too frequently is encouraged to [contact us](#) immediately by email.

The final judgment as to what constitutes overuse of the classifieds is the sole judgment of MoHPC. There is no appeals process. Belligerent, insulting, and/or demanding emails or postings will be counter-productive. MoHPC will take whatever steps it feels necessary to keep the Classified Ads section a friendly place where buyers and sellers are on equal footing.

If you have an ad for a service or a product that doesn't immediately sell, post an ad no more frequently than once per week. Please proofread, and spellcheck before posting, and use the preview button to check your formatting as the current software does not allow editing after posting. To be clear, particular *items* should not be advertised more than once a week, regardless of whether those items are advertised individually or as part of a larger ad.

Do not advertise replica/reproduction items that replicate a trademark unless you have written permission. Any other replica/fake materials must be clearly indicated.

Be careful to use the correct heading for your ad. For example do not use (FS) if you are selling by auction or For Sale Best Offer.

Advertisements are submitted by individuals who are likely to be strangers to the Museum curator. Treat advertisements here like advertisements anywhere else on the net and take precautions that you feel are appropriate.

[Go back to the main exhibit hall](#)



Formatting

It's easy to add formatting to posts in the HP forums (except for the classified ads). You simply type codes that look like [code]. Codes come in two types - single and paired. Paired codes have a code that turns them on like [bold] and one that turns them off like [/bold]. These codes are very similar to HTML.

Line Breaks and Program Listings

By default the forum software flows your text to match the boundaries of the browser. When you type two blank lines, you'll get a new paragraph with a blank line in between. To get a single line break without a blank line use [nl] which stands for "new line". To keep a block of text looking just like you entered it, surround it with [pre] and [/pre]

First[nl]Second produces: First
Second

[pre]The first line
second line
third line
fourth line
etc.[/pre] produces:
The first line
second line
third line
fourth line
etc.

Links and Images

- [link:http://www.hpmuseum.org]Link to Museum[/link] produces [Link to Museum](http://www.hpmuseum.org).
- [url:http://www.hpmuseum.org] produces <http://www.hpmuseum.org>.
- [img:http://www.hpmuseum.org/gto.gif] produces 

You need a full URL/address immediately after the colon. The only difference between [link] and [url] is that [link] allows you to specify the text that the user sees.

Text Formatting

- [bold]bold text[/bold] produces **bold text**
- [italic]italic text[/italic] produces *italic text*
- [un]underlined text[/un] produces underlined text
- [super]text[/super] produces ^{text}
- [sub]text[/sub] produces _{text}
- [big]text[/big] produces text
- [small]text[/small] produces text

Quoting Someone

[quote]Some text you are quoting[/quote] Your response... produces:

Quote:

Some text you are quoting

Your response...

It's usually not necessary to quote but if you want to reply to just part of a long post, you can cut and paste the appropriate section(s) into your response and surround it(them) with [quote] ... [/quote].

Lists

[ol][li]an item[li]another item[/ol] produces: 1. an item
2. another item

[ul][li]an item
[li]another item[/ul] produces: • an item
• another item

Note the the list is started with [ol] (ordered list) or [ul] (unordered list) and ended with the matching end code. Each list item is started with [li] and there is no end code.

Headings

[head]Headings[/head] produces the heading directly above.

Miscellaneous

- [center]centered text[/center] centers text on the page
- [hr] means horizontal rule and draws a line across the page

Leaving []s alone

If I want to explain formatting in a post you can "escape" codes by preceding the brackets with backslashes. For example: \[pre\] will produce [pre]. In cases where you enter something in brackets that isn't a format code, the system will automatically escape the brackets for you. For example you will get the same effect entering \[u\] or just [u]. In both cases [u] will be displayed in your message.

Examples:

Input:	Results:
[img:http://www.hpmuseum.org/gto.gif]	
[url:http://www.hpmuseum.org/hp65.htm]	http://www.hpmuseum.org/hp65.htm
[link:http://www.hpmuseum.org/hp65.htm] HP-65 Page[/link]	HP-65 Page
Next is an [head]Article Heading[/head]	Next is an Article Heading

for the next bunch of text	for the next bunch of text
Here are some lines separated by one newline[nl]line1[nl]line2[nl]line3[nl]line4	Here are some lines separated by one newline line1 line2 line3 line4
Note that text separated by one line break, like this, gets merged into a paragraph.	Note that text separated by one line break, like this, gets merged into a paragraph.
If you want to do a code listing:[pre]preformatted textline 2line 3line 4line 5[/pre]	If you want to do a code listing: preformatted text line 2 line 3 line 4 line 5
Here's an ordered (numbered) list[ol][li]First element[li]Second element[/ol]	Here's an ordered (numbered) list 1. First element 2. Second element
Here's an unordered (unnumbered) list[ul][li]First element[li]Second element[/ul]	Here's an unordered (unnumbered) list • First element • Second element
Here's a [super]superscript[/super] and a [sub]subscript[/sub] and a [bold]bold [super]superscript[/super] and [sub]subscript[/sub][[/bold]	Here's a ^{superscript} and a _{subscript} and a ^{superscript} and _{subscript}

You'll be able to see the results by pressing the **preview button** rather than the post button. If the results don't look right (for example if you see something like: [bold]bold text[/bald] in the preview screen, go back and correct the error and then preview again.

More Details

Always use a full URL such as <http://www.hpmuseum.org/enter.jpg>. Note that URLs can not contain any of the characters <, >,], ", ' and spaces.

If you enter a code incorrectly and post it, you can fix it by pressing the edit button on your post. You may need to remove backslashes when you edit. For example if you enter [bold]test[bold] and then edit, it will come back as \[bold\]test\[bold\]. This is because the software can't tell whether you made a mistake entering [bold]text[/bold] or whether you really meant it to display like that to show someone how NOT to enter codes. In this case, delete the escapes and enter the slash on the bold end code so you have: [bold]test[/bold] and then press save. If you enter something like a[i] = b[k], it will look the same when you edit it because there are no codes [i] or [k] so there is no need to escape them.

If you don't have a place to host your images, then please [log in](#) and [request a guest directory](#). Once the directory is

created, you will be able to upload your files. (This is actually preferred since I won't have to worry about the images moving or disappearing in the future and leaving broken links.)

 [Go back to the main exhibit hall](#)

 [Go back to the articles forum](#)

 [Go back to the HP forum](#)

HP Articles Forum

[Return to the Index]

[Previous]

HP-41 MCODE: The Last Function - at last!

Posted by [Ángel Martín](#) on 8 Nov 2013, 4:55 a.m.

The Last Function - at last!

The latest releases of the PowerCL, SandMath and SandMatrix modules all include support for the LASTF functionality. This is a handy choice for repeat executions of the same function (i.e. to execute again the last-executed function), without having to type its name or navigate the different launchers to access it.

The implementation is not universal - it only covers functions invoked using the dedicated launchers, but not those called using the mainframe XEQ function. It does however support two scenarios: (a) functions in the main FATs, as well as (b) *those sub-functions from the auxiliary FATs*. Because the latter group cannot be assigned to a key in the user keyboard, the LASTF solution is especially useful in this case.

The following table summarizes the launchers that have this feature: (lower case s denotes the SIGMA character)

Module	Launchers	LASTF Method
Power_CL rev. BS14	SCL, XCAT, TURBO, BAUD, MMU, IMDB, HEPX, YBUF, XSEQ IOBUS, IOPG#, STR\$, ALP\$ XQ1 _ , XQ2 _ PLUG, XFAT , XROM	Captures (sub)fnc id# Captures (sub)fnc id# Captures (sub)fnc NAME Not supported
SandMath 3x3 rev. 4K	SFL, HYP, FRC, RCL# SFL\$ _ SFL# _ _ _	Captures (sub)fnc id# Captures (sub)fnc NAME Captures (sub)fnc id#
SandMatrix rev. 4M	SML, SDST SVL\$ _ SVL# _ _ _	Captures (sub)fnc id# Captures (sub)fnc NAME Captures (sub)fnc id#

Note that the launchers XQ1, XQ2, sFL\$ and sVL\$ will switch to ALPHA mode automatically.

Operating Instructions

The Last Function feature is triggered by pressing the radix key (decimal point - the same key used by LastX) while the launcher is up. This is consistently implemented across all launchers supporting the functionality in the three modules - all work the same way.

When this feature is invoked, it first shows "LASFT" briefly in the display, quickly followed by the last-function name. Keeping the key depressed for a while shows "NULL" and cancels the action. In RUN mode the function is executed, and in PRGM mode it's added as a program step if programmable, or directly executed if not programmable.

If no last-function yet exists, the mainframe function XEQ is parsed in the SandMath and PowerCL modules; whilst

the error message "NO LASTF" is shown in the SandMatrix module. If the buffer #9 is not present, the error message is "NO BUF" instead in all cases.

Implementation details.

The functionality is a two-step process: a first one to capture the function id#, and a second that retrieves it, shows the function name, and finally parses it. All launchers have been enhanced to store the appropriate function information (either index codes or full names) in registers within a dedicated buffer (with id# = 9). The buffer is maintained automatically by the modules (created if not present when the calculator is switched ON), and its contents are preserved while it is turned off (during "deep sleep"). No user interaction is required.

The buffer has one register reserved for each of the three modules, so that up to three last-functions are simultaneously available. A total of five registers are used, as follows:

Register	Used for:
b4	SandMatrix fcn id#
b3	SandMath fcn id#
b2	PowerCL fcn id#
b1	Time-based Seed
b0	Buffer Header

The fcn id# format can be either a string of alpha characters (stored in reversed order) representing the function name, or a three-digit hex value representing the sub-function index. In the latter case the [MS] field is marked with an "F" to tell the cases apart.

When the LASTF action is triggered pressing the Radix key, the code seeks for the function data in the relevant buffer register, depending on the module carrying the action. When found, it displays the sub-function name (either by mirror-imaging the id# or by looking it up in the corresponding auxiliary FAT. Finally it's parsed to the corresponding section responsible for the execution of the sub-functions.

Dependencies

These three modules do require the 41-CX OS and the Library#4 installed. Note that the Library#4 was also updated to revision "K" - now including subroutines in support of this functionality. As always, make sure that matching revisions are installed in your machine.

Note as well that the SandMatrix also requires the SandMath installed.

Edited: 8 Nov 2013, 5:19 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

HP-80 History, Design and Interesting Facts

Posted by [Max Stone](#) on 24 Oct 2013, 4:14 p.m.

The World's First Businessman's Pocket Calculator





Introduction

Hewlett-Packard redefined portable computing with the launch of the HP-35 in 1972. This was the world's first pocket calculator with transcendental functions (sin, log, etc.) and the first with RPN (Reverse Polish Notation). Overnight, the slide rule was dead.

A month after the launch of the HP-35, Bill Hewlett had breakfast with 200 securities analysts in New York. HP had placed HP-35s at each breakfast table and Bill noticed that "...although they were familiar with electronics, it was evident that such key designations as sin, cos, log and e^x were more a barrier to an understanding of the calculator's function than a help...Right there and then it became evident that what was needed was a pocket-sized calculator designed specifically for the business community..."

Bill realized that there was a gap between the language of the financial user and their development engineers, so he put Bill Crowley from their corporate finance department together with France Rode, from HP Labs, who was an expert in programming and assigned the project leader of the HP-80 development team. The two visited a number of potential business users and from their discussions identified a basic set of financial problems that were eventually incorporated into the HP-80. HP had to develop many new financial algorithms to deal with the key-entry logic, design architecture and limited memory of the device.

Head of R&D and instrumental in the development of the HP-35, Barney Oliver helped arrange the keyboard to allow logical entry of financial problems. With over 30 financial functions, it was necessary to provide shifted functions on several keys, so the HP-80 became HP's first calculator to feature the now famous unlabeled gold prefix key that continues on HP calculators to this day.



In four short months, HP had locked-down the details of the Model 80 which was essentially a modified HP-35 sharing the same

architecture, but now with a total of seven ROMs (up from three) for the microcode and the higher-level functional implementations. The seven ROMs were mounted on a hybrid circuit to form a single assembly, with each ROM containing 256 ten-bit words, giving a total ROM memory size of approximately 2Kb. (Image above shows the hybrid chip with the seven ROMs in a row. These are protected under a metal cover in production.)

By mid-December HP had started limited production. The HP-80 was unveiled to the world's media on 16 January 1973 at 15 press conferences around the world. The price for the HP-80 at launch was \$395 and was supplied in a protective case that held the calculator and accessories (leather case, AC adapter and manual).

Several versions of the HP-80 were released over its lifetime but these were mainly minor cosmetic variations with the exception of the final production version that used quad ROMs (two HP-45/55/70 style DIPs) rather than the seven ROM hybrid. Unlike the HP-35, the HP-80 had no significant bugs when it was first released. The HP-80 was discontinued in 1978, replaced by the HP38E introduced that same year. During its lifetime, the HP-80 witnessed the launch of the remaining 'Classic' series calculators, the second generation ("Woodstock") and many of the third generation ("Spice") calculators.

There are many HP-80s still in circulation, even after being discontinued 35 years ago. This document is a detailed (but not exhaustive) overview of the HP-80 including photographs from an example I recently acquired.

What's in the Box?

The first HP-80's were supplied in a vinyl-covered metal carry case with snap closure. The plastic liners have a type of felt coating with moulded pockets and cut-outs for storing the calculator and accessories securely.

The 78-page Owner's Handbook is stored in the lid pocket and has a pocket in the inside back cover for the Service card and Addenda card. There is an Accessories brochure that lists optional accessories for the HP-35 and HP-80 including the security cradle, field case and battery holder (charger). There is also a 20-page spiral-bound Quick Reference Guide that fits inside a pocket of the separate leather case. The battery charger/AC Adapter is switchable between 110V and 220V and has the part number 82002A.

Later HP-80s were supplied with a plastic carry case and a synthetic leather calculator case. This example is also the early version of the HP-80 with only the words "Hewlett-Packard" on the bottom edge. Later versions added the model number "80". The OFF/ON wording next to the power switch was changed from printed text (as here) to raised molded letters in later versions.

Exploring the Design and Internals of the HP-80 Calculator

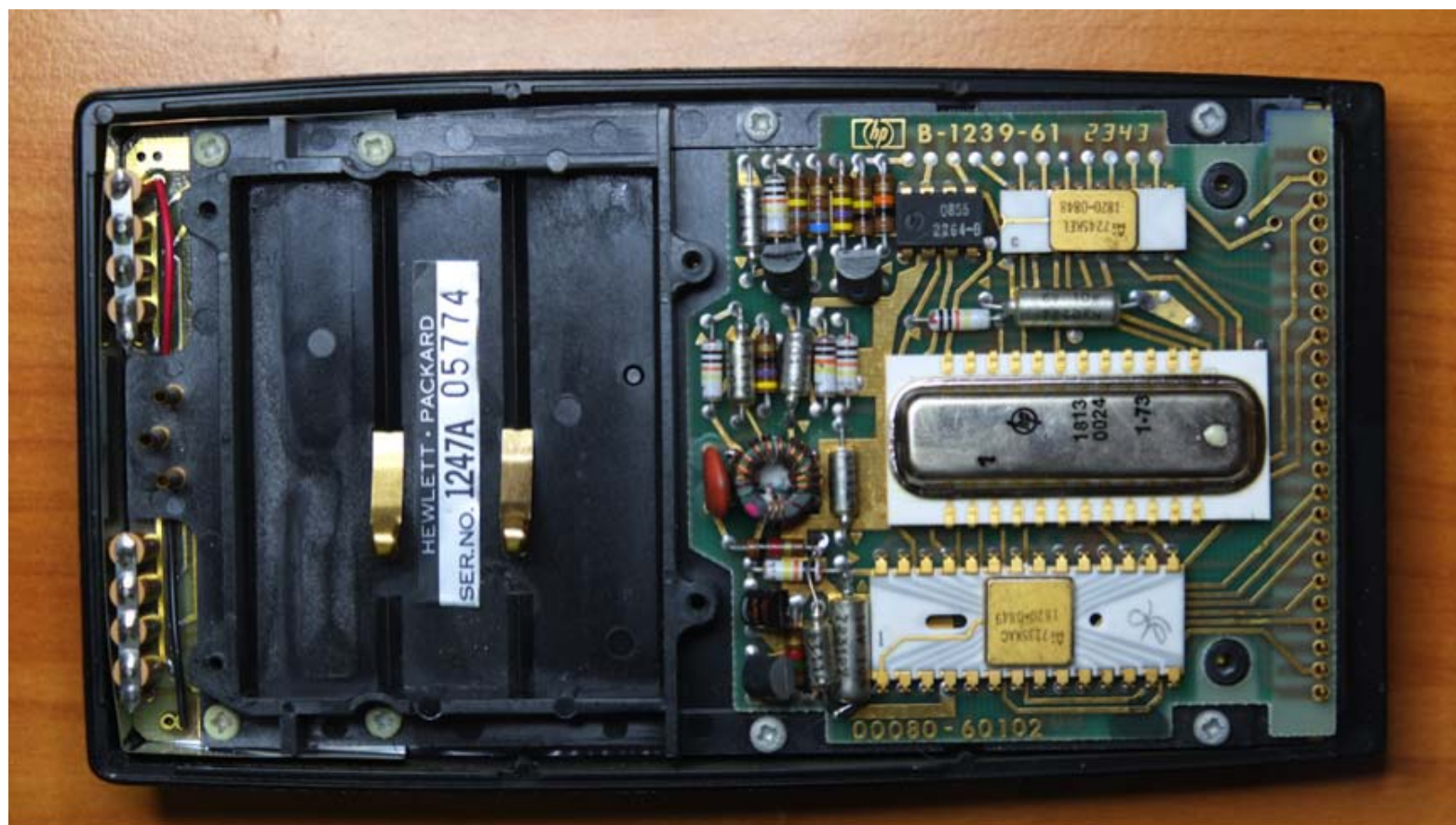
The serial number of this example is 1247A05774. While this suggests a production date of December 1972, HP's early calculators used the first four digits to track version numbers indicating that this example was from the earliest product batch that started at the end of 1972. The opened calculator reveals a production date around January 1973.



Like all Classic series calculators, the HP-80 is quite easy to open. There are six screws holding the case together. Two are inside the battery compartment, two are below the bottom rubber feet and two are below the top edge of the rear label. Getting to the screws below the label requires that the label is carefully peeled back. The label is thin aluminium and it can be gently lifted and pulled back from a corner if the glue is released with some heat from a hairdryer. It's nearly impossible to avoid some creasing of this label so you must accept some imperfections when pressing it back into place. The right screw is just below the 'S' of the word

“INSTRUCTIONS” so only about one quarter of the label needs to be peeled back.

All the fasteners used in the HP-80 are zinc plated self-tapping steel screws and all had a fine white crystalline coating on the exposed head from 40 years of airborne contamination. The deposit easily rubs off but leaves the screw head a dull color.



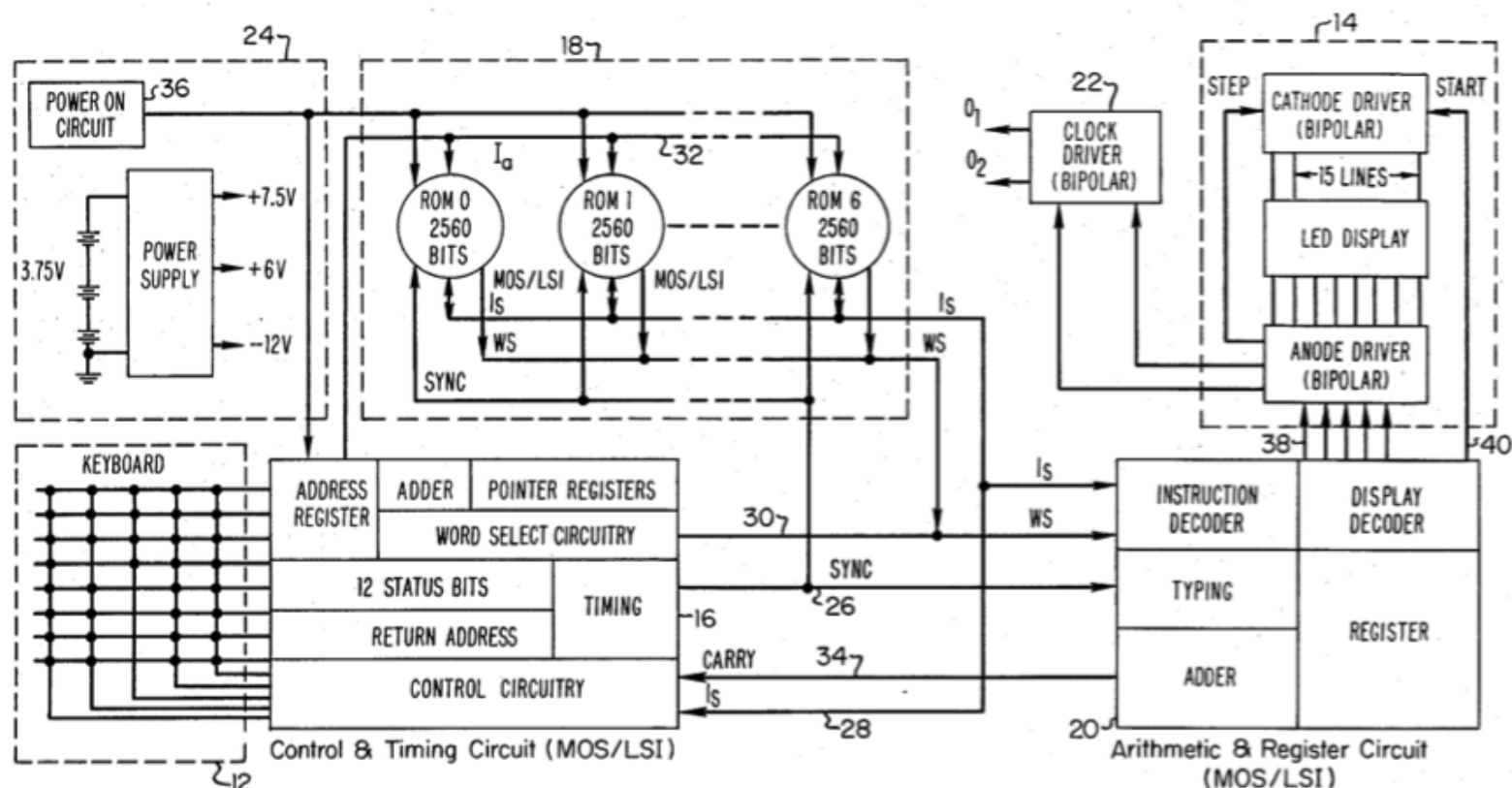
The image above shows the calculator with the back removed. The two sets of 4 passive components on the far left are coils used in the inductive circuit to drive the LEDs. This is a different approach where the current driving the LEDs is limited by the time it takes to build up in the inductor, rather than through a resistor in conventional LED displays. This strobed inductive drive circuit is more efficient and was invented and patented by HP for use in calculators. In the worst case, display power consumption is about 110 milliwatts (thirteen 8's and two minus signs). The LED display is covered later in more detail.

The connector for the AC adapter is between these coils. The red and black wires connect the battery connectors to the circuit board. There is a metal-sprung connector held in the back cover (not visible here) that acts as a switch when connecting the AC adapter by isolating the battery circuit from the calculator (to allow charging) when the connector is inserted. This is why the HP-80 won't run on if the AC adaptor is plugged in but off at the wall socket. Although gold-plated, the metal-sprung connector can corrode with the result that the connection fails and the battery cannot power the calculator. This example had unreliable operation when on battery power as a result of corrosion at this location, with the gold-plating flaking off the metal.

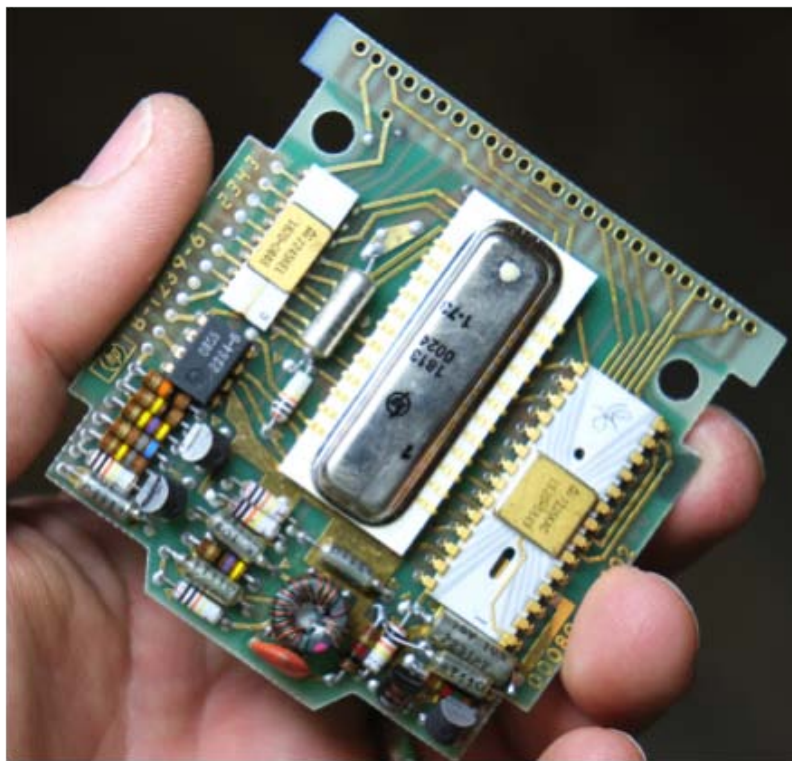
The HP-80 does not have a single CPU but contains five MOS/LSI (metal-oxide-semiconductor, large-scale-integration) circuits made up of seven read-only-memories (ROMs), an arithmetic and register circuit (A&R), and a control and timing circuit (C&T). The logic design was done by HP and the circuits were developed and manufactured by two outside vendors. Three custom bipolar circuits were manufactured by HP's Santa Clara Division: a two-phase clock driver, an LED anode driver/clock generator, and an LED cathode driver. (The block diagram below was taken from HP's 1972 patent document and shows these circuits).

The three MOS circuits are two-phase dynamic MOS/LSI circuits with low thresholds allowing compatibility with TTL bipolar circuits and allowing extremely low-power operation (less than 100 milliwatts for all three MOS circuits). They are organized to process 14 digit BCD (binary-coded decimal) words in a digit-serial, bit-serial manner. The maximum bit rate or clock frequency is 200 kHz, which gives a word time of 280 microseconds, permitting a floating point addition to be completed in 60 milliseconds, or about 17 a second. (A 3 GHz Pentium 4 can do about 3 billion additions per second!)

In total, the LSI P-channel MOS circuitry in the HP-80 has about 40 thousand transistors. This is a little more than what Intel squeezed into the original 8086 in 1976, but a tiny fraction of the 1 billion transistors Intel now have keeping close company in the Core i7 chip.



The HP-80 uses two printed circuit boards for all the circuits shown above. There is a larger board that carries the keyboard connectors, the LED display, drivers and power connectors. The smaller circuit board (shown below) carries all the MOS logic, clock driver, and the power supply components. This is the "heart" of the HP-80 and a masterpiece of 1970's electronics design.



The seven ROMs are under the metal cover in the center of the circuit board (1813-0024). The passive components and transistors make up a power on signal circuit and DC to DC converter for the three supply voltages the calculator needs. The large white chip is the Control and Timing circuit (1820-0849), the smaller white chip is the Arithmetic and Register circuit (1820-0848) and the black chip is the bipolar clock driver. The row of connectors at the top connect to the keyboard circuit board that is sandwiched between a plastic internal frame and the front half of the case, and secured by the 8 screws.

Once opened you can see that while the serial number indicates a December 1972 batch number, the ROM chip is dated January 1973. Production dates are difficult to determine accurately as HP used pre-printed aluminium serial labels that started with 1247A (or S) for the early batch runs in America and Singapore, until the following 5 digits were no longer sufficient. Accurately dating these earlier versions is only possible by taking the calculator apart.

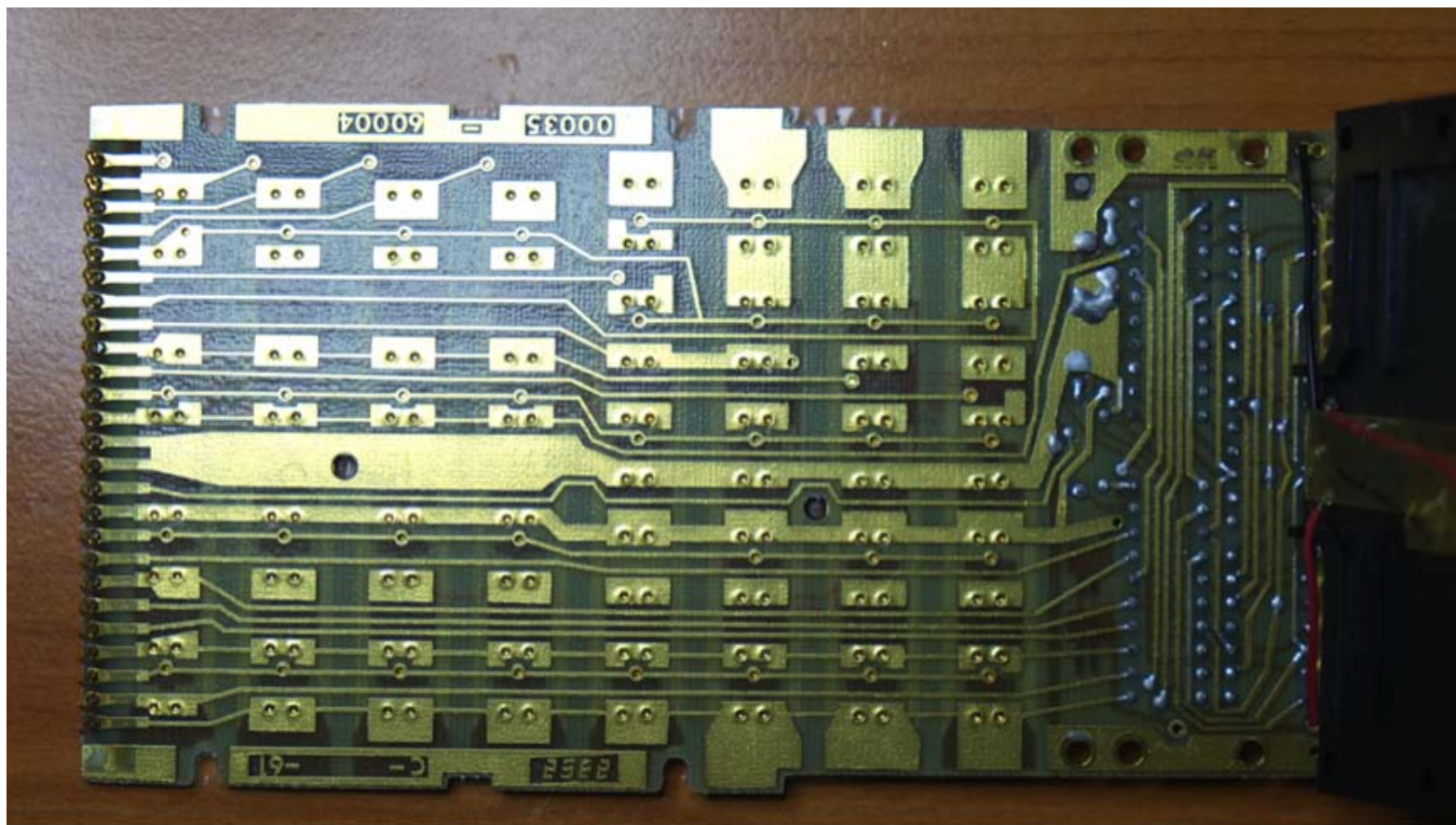


With the main circuit board removed, you can clearly see the connectors from the keyboard circuit board. Each is a tiny split spring that clips into its corresponding hole in the main processor board. There is no solder or latching mechanism and the main processor board is held in place by these connectors and two locating holes that match the two lower screw bosses.

The 8 screws holding the internal plastic frame also had surface corrosion, but like the external case screws, the white powder was easy to clean off. With these screws removed, the plastic frame and keyboard circuit board is released from the front half of the

case. These assemblies are attached via wires leading from the battery and AC power connectors, visible at the right of the following image. The red leads terminate on the left and right pins of the AC adaptor connector, closed by the metal-sprung switch when the calculator runs on batteries. Clear insulation tape covers the soldered wires of the battery connector to prevent shorting on the circuit board.

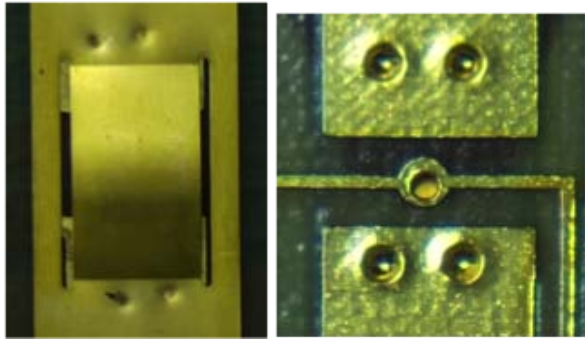
The picture below shows the underside of the keyboard circuit board:



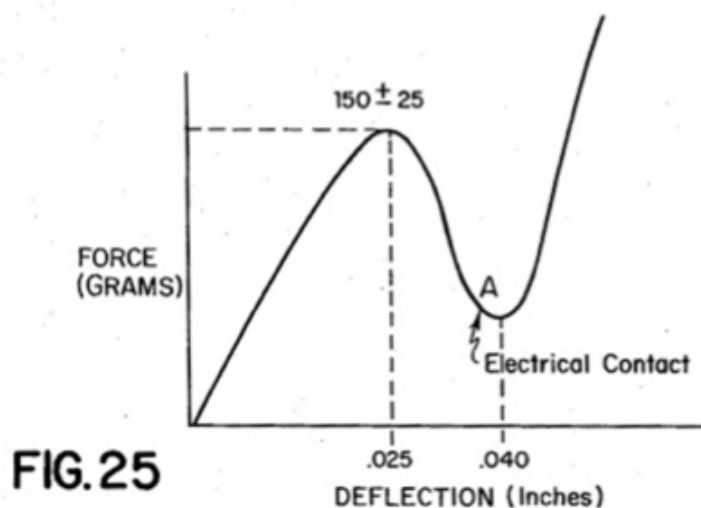
The picture below shows the keyboard circuit board, LEDs and related drivers. Also visible is the clear plastic layer that protects the internals from spills and dust that could otherwise enter the calculator around the keys. Also note the black rubber strip immediately below the LEDs. This strip prevents unsightly reflections off the pins below. I removed this strip for the remaining photographs.



The keyboard mechanical design of the HP-80 is identical to that designed for the HP-35, and patented by HP in 1971. HP calculators are famous for their keys, so they're worth a closer inspection.



In their patent submission, HP describe the keyboard as a “reliable, low-profile, low cost keyboard with tactile feedback”. There are eight metal strips that run across the keyboard. Each of these strips has 4 or 5 small rectangular ‘humps’ less than 1mm high corresponding to each key, made by deforming (stretching) the metal in a shaped press. These strips are spot-welded to the circuit board and there are traces that run at right-angles spot welded to contacts that sit directly below the curved hump. Pressing the key makes contact allowing the calculator to identify the key by its grid location.



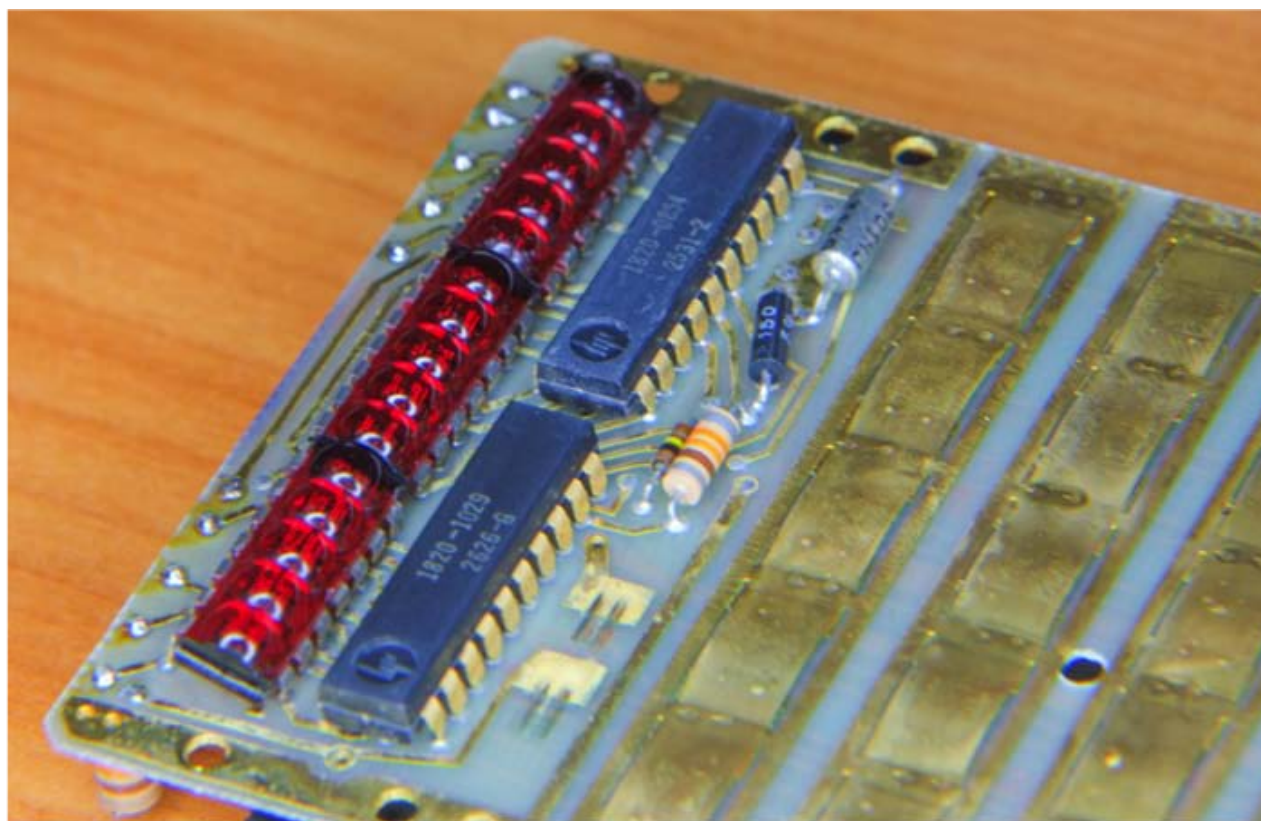
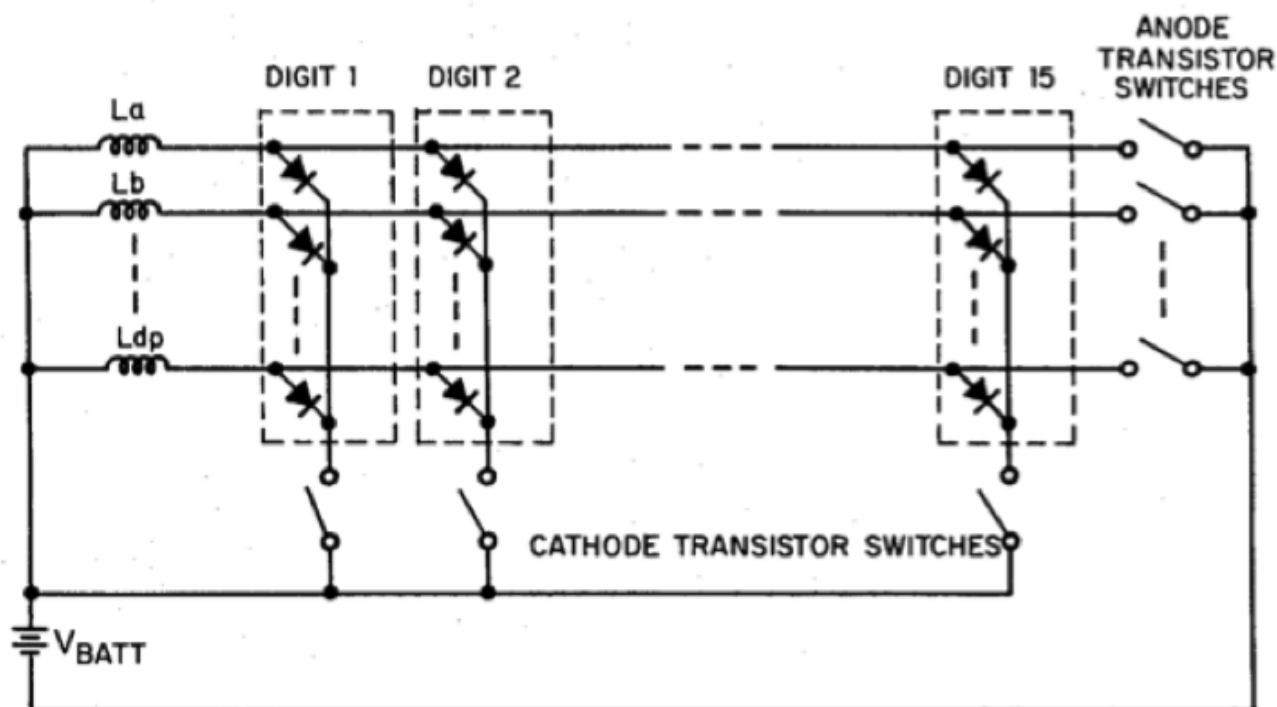
Mechanical bounce is less than one millisecond and the calculator contains a wait loop to prevent double entries. HP tested the design extensively and rate the design at over one million cycles. Critical to the success of this system and the overall feel of the key press, is the way the key has an “over-center” or “fall away” feel. When pressing the key, the operator must exceed 125 grams of pressure at which point the metal hump “breaks” through and electrical contact is made. The operator cannot prevent contact from happening once the break-point is reached, and when releasing the key, there is again a release-point when the operator cannot prevent the key from breaking contact. This patented design gives HP keyboards their distinctive feel and prevents missed or multiple entries. Even 40 years later, the keys have a crisp, accurate action because of this excellent design.

The final mechanical design feature worth mentioning is the way each key is hinged on the lower edge giving a rotational rocking motion when pressing the key. There is a clear plastic frame that fits below the entire keyboard and keeps each hinge located in the keyboard surround.



One of the biggest challenges in the HP-35 project was the design and implementation of the LED display. The HP-80 uses the identical LED design to the HP-35, as with the rest of Classic series of HP calculators.

The HP-80 uses three magnified five-digit clusters, saving both power and cost. Energy is stored in inductors and dumped into the light-emitting diodes using a pulse drive circuit. The LED display is made up of a 8×15 array of individual LED segments, 8 segments per digit (7 for the digit plus a decimal point) and 15 digits. Eight rows are scanned by the Anode driver (1820-1029) while the 15 columns are scanned by the 15 position shift register Cathode driver (1820-0854). The scanning operation moves from digit to digit in response to counter clock signals from the Anode driver.



The Anode driver circuit includes an LC circuit (resonant circuit) to set the scanning frequency of the display, made up of the black 150pF capacitor and 330 uH inductor to its left. The small resistor above the inductor trims the reference voltage for detecting low power and turning on all the decimal points. The silver component to the right is a 2.2uF capacitor used to stabilize the power used by the LC circuit. The LC circuit generates a frequency of approximately 800 KHz, divided by 4 to 200 kHz by the flip-flop stage of the Anode driver circuit, and used to time the scanning of the display.

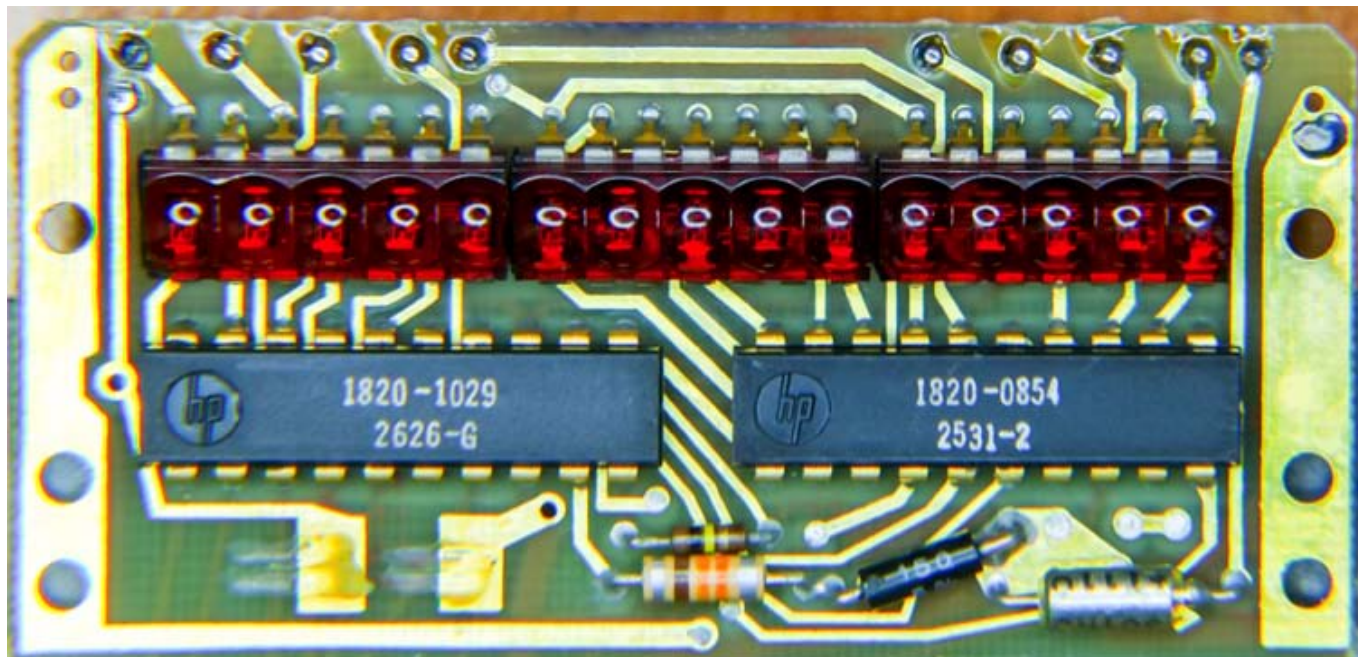
The solder points at the top of the board connect the 8 inductors used to drive the pulsed display circuit. Each inductor has a value of 130uH except the 68uf decimal point. The LED segments that make up each digit are pulsed individually, sweeping each digit

sequentially and constantly refreshing the entire display at a speed too fast for the eye to see. (The blank delay when pressing a key is the display wait loop routine).

The display has 15 digits however the calculator's word cycle is only 14 digits long. The extra character is the decimal point which is a special case as its position is generated by the Arithmetic and Register circuit or by the low voltage comparator in the Anode driver circuit that can trigger all decimal points to light up (indicating a low battery).

The three cluster of five LED digits each on the dual in-line package (DIP) in these images, with the moulded magnifying lens above each digit. Each DIP has 14 legs, connecting 8 anodes and 5 cathodes (one pin not connected internally). The common anodes across all three clusters are wired together while the 15 Cathodes are connected individually to the Cathode transistor switches.

The power switch contacts are visible below the Anode driver MOS chip. Even after 40 years, the grease on the board has not dried and the switch slides smoothly.



The first digit in the display is only used for the minus sign. The following 11 digits are used to display the mantissa and the decimal separator which occupies the space of an entire digit. The 13th digit displays the sign of the exponent and the last two digits display the two digit exponent:

-1.234567899-99

Later HP LED calculators had less characters in their displays. The Woodstock series had 12 digits however space was saved as the exponent shared space with the mantissa and the decimal point shared space with a displayed digit. The last LED-series calculators added a comma segment. HP launched their first LCD calculator, the HP-41C in 1979 and by 1983 HP had discontinued all their LED pocket calculators in favour of LCD models.

The Battery Rebuild

After 40 years the original NiCad battery pack was shot and was beginning to leak corrosive electrolyte. Fortunately the battery contacts of the Classic series are not at the ends of the battery but midway down so damage to the contacts can be avoided if the battery is not leaking too badly.

I split open the plastic carrier with a craft knife and replaced the three failed NiCads with new NiMH AA rechargeables (2500 mah), salvaging what I could from the original connectors to reuse. I used the hot soldering iron to melt closed the plastic shell again. NiMH batteries have similar charging and discharging characteristics so there's no problem adopting this newer technology as I'll get about twice the battery life compared to the original NiCads, although it may well take longer to fully charge these batteries using the standard HP AC adaptor, and long-term electrolyte leakage is still an issue.

I have ordered a rebuilt battery back as my rebuild is a bit unprofessional.

Final Thoughts

It's clear that HP put a huge amount of effort into the original development of the HP-35 calculator. The term 'over-engineered' springs to mind, especially when you consider that these Classic series calculators can survive decades of use.

How to get another 40 years of life? I'd definitely replace the 14 zinc plated steel screws used to hold everything together with stainless equivalents, as it seems that just long-term atmospheric exposure can corrode these.

A more extreme upgrade would be to replace the three 1.2V NiCad or NiMH AA batteries with three 3.6V Lithium Ion AA batteries wired in parallel inside the battery pack, and rewire the AC cable onto a suitable Li-Ion charger. This will banish the leaking battery issue forever, but I'll need to investigate the viability of this upgrade further.

It's interesting how the HP-80, lacking programmability and being HP's second pocket calculator gets far less attention than the HP-35, but they really are brother and sister and along with the rest of the Classic series of calculators, deserve all the recognition we can give to these little marvels of 70's engineering.

End Note - Bugs?

While the HP-80 had no serious bugs, I have identified a bug that manifests as a strange characteristic when in exponential display mode with a positive number less than 1, and CHS is pressed. Instead of only inserting a negative sign in the left-most position, it also changes the exponential value to -99.

Toggling CHS leaves the exponential value at -99 but toggles the left-most negative sign as expected. However, pressing SAVE resets the exponential to the original value, switching back to decimal display shows the correct value and calculations are correct even when the exponential value shows -99.

This example keystrokes sequence and display shows this bug on the HP-80 (at least with the seven ROM hybrid version):

Power on " 0.00 " (default display at power on)
.1 SAVE " 0.10 " (enter 0.1 on stack)
Shift 7 " 1. -00" (switches display to scientific notation)
CHS "-1. -99" (Here's the bug, as should just add the leading negative)
Shift 2 "-0.10 " (switches back to decimal notation, shows the correct value)

I happened to stumble upon this bug trying to get both the negative exponent and leading negative in the display for the beauty shot at the beginning of this blog.

(Kind thanks to Katie Wasserman and Eric Smith for confirming this bug on other HP-80s and confirming it's likely a firmware bug in the HP-80 hybrid ROM version.)

Acknowledgements: All photographs are my own except the black and white image of the seven ROM hybrid chip taken from the May 1973 edition of the Hewlett-Packard Journal which covers the HP-80 in details. The diagrams are taken from US Patent 3,863,060 filed Oct 1972 and described as "General purpose calculator with capability of performing interdisciplinary business calculations". This 80-page document is essentially a detailed specification of how the HP-80 works, right down to the binary coding of the seven ROMs. The rest of the information comes from various internet sources too many to reference, so I make no apologies for errors or omissions.

Original blog posting can be found here: [HP-80 Blog Entry](#)

Edited: 24 Oct 2013, 4:33 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

RPN-1200

Posted by [Benoit Maag](#) on 25 Aug 2013, 1:02 a.m.

Though I became a HP fan shortly after, my first calculator was actually a TI-1200, a basic calculator with the 4 operations and % with a very nice design. My father's company had given calculators to their best customers, and some had been returned as 'defective'. All it took was to replace the 9V battery and there I had my first calculator.

So just for the fun of it, and in admiration of the equally zen design of the Commodore 7919 (scientific model with just 19 keys), I endeavored to design a Microchip PIC based board for TI-1200 that would convert it to a scientific RPN model.

So here it is: RPN, trig/log/power functions, metric<->US conversions, 4-level stack with LASTX, input through a 'command line' with <- capability, 10 memories, store and recall arithmetic, FIX and SCI modes.

The code is written mostly in C (some inline ASM) using the Microchip MPLAB X IDE. The only drawback is that the floating point routines are binary based and the precision is really not good... If anyone has worked on a similar project, or has recommendation / source code for BCD arithmetic and floating point functions in C or PIC assembly, I am definitely interested.

Pictures of the RPN-1200 vs TI-1200 and of the inside of the RPN-1200 are at:

<http://www.hpmuseum.org/guest/benoitm/rpn1200.jpg> <http://www.hpmuseum.org/guest/benoitm/inside.jpg>

Thank you,

Benoit

Edited: 27 Aug 2013, 12:41 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

41 SOLVE & INTEG Bank-switched Implementation.

Posted by [Ángel Martín](#) on 21 June 2013, 11:12 a.m.

Programming highlights: MCODE Cathedrals.

Often when visiting a landmark or a commemorative building we feel the imposing presence of something that's bigger than what any possible description could convey, and we proceed tip-toeing, speaking in whispers not to disturb the spirit of its creators... this is exactly how I feel about the addition of the SOLVE and INTEG "cathedral of MCODE" to the SandMath!

Leaving the mathematical prowess and attributes aside - as tremendous as they are - the housekeeping chores and implementation on the 41 platform are nothing short of spectacular. The original programmers we're told adapted the already-existing code from the HP-15C, but they had to overcome a couple of real challenges to port it smoothly into the 41 platform. Possibly the 15C also required similar trickery, but I don't know its internal structure so I can't say.

The first striking thing is of course being able to return to an MCODE code stream after executing a user code program (FOCAL) - which calculates $f(x)$. That alone can leave you thinking - as it did to me, suspecting the hows and the abouts being explained somehow by SIRTN and SILOOP, the two auxiliary functions written for this purpose. - Yet how did they exactly work? We need to understand the buffer-14 paradigm before we can answer this.

The unsung Hero

Yes, there's the question of Buffer-14, the dedicated buffer in the Advantage that exhibits a rather idiosyncratic temperament: contrary to all other modules, the Advantage seems to be on a "search and destroy" mission, with the apparent aim to kill any previous existence of the buffer, judging by the polling events CALC_OFF and IO_SRVC.

Equally intriguing is the location of said buffer, which is situated (while it's allowed to exist) below the Key assignments area - and not above it as it's the normal way. This fact conflicts with the OS routines that manage the I/O area, like [PKIOAS] and others, and would create real havoc if it weren't because the Advantage manages the buffer dynamically, creating it on-the-fly just when the execution starts, and killing it upon termination. So as far as the rest of the machine is concerned (OS included), it is as if buffer-14 had never existed!

But why all that hassle, you'd ask? Couldn't they have used the normal approach to hold whatever data that needed to be stored in a standard-type buffer, like every other implementation does? I believe the reason was to have an absolute location for the buffer registers: with the starting location for buffer-14 always being 0x0C0 (192 dec) the access and retrieval of the values stored there becomes a much easier affair, just using their fixed "register numbers". This may have made using the 15C algorithms simpler, and avoids altogether the relative addressing problem present when the buffers are placed in their "regular" space (which incidentally I became very aware of while writing the 41Z complex stack buffer implementation).

However one of the implications of wedging a buffer below the key assignments area is that the code would first need to move them all - as well as all other buffers already present - up in memory, to make room for the newcomer. And

conversely, this will have to be undone upon termination of the function execution.

Now you can imagine the housekeeping chores required, and the intricacies of the implementation in the code. That's why the IO_SRVC event is constantly checking for the presence of buffer-14, proceeding to its removal if found at a non-suitable time.

Let's add to this mounting MCODE nightmare the requirement that both SOLVE and INTEG would work in a nested way, which is something that the code will only discover having already created the buffer for the first function – so the buffer would have to be resized on the fly, not losing any previous information already contained.

Entering the Paralell Dimension.

And adding insult to injury, welcome to the parallel dimension of bank-switching: imagine now attempting to do all that from within an auxiliary bank (say bank-3), which when activated would not know a thing about the main one (little details like the FAT, etc); so it's there to live and die by its own sword. Case in point being: what if the function $f(x)$ to solve or integrate contains functions available within the same module, how then could they be found?

Well at least this one has an easy answer: bank-1 needs to be the active one while the FOCAL program runs, thus obviously the main FAT is also there and all will work out. So provided we can identify the exact points in the code where the execution is transferred to the FOCAL label we'd be home free, or would we? But beware, because then not only the MCODE execution needs to be resumed (how is still pending clarification), but it'll also have to re-activate bank-3 as the very first thing it does.

Buffer-14 comes to the rescue.- Say there are two auxiliary functions, one of them SIRTN is sought for during the initialization, and its address is placed in the RPN return stack, just above the other address for the global LBL that calculates $f(x)$. This will ensure that SIRTN will run after LBL $f(x)$ is finished, ok so we've got control back – what to do with it? Say now that the second auxiliary function SILOOP is the very first (and only) line in SIRTN, that'll send the execution back to our MCODE – way to go, but this is a new function that has no recollection of the past or knows nothing about whatever was done before, unless...

Unless of course we use the buffer as data structure to do the parameter passing! Isn't this brilliant? Yes of course, that's the answer: SILOOP will retrieve the necessary information from buffer-14 to resume, picking up exactly where it was left off prior to calling LBL $f(x)$. Mind you. It'll also have to make sure things are as expected when it "wakes up": is the buffer there, which function was run (SOLVE or INTEG), and react adequately if some of the information is not there. This can happen if a user programs SILOOP inadvertently, of course (although they could have make it non-programmable I suspect they didn't care anyway).

The last refinement included to speed up things was to also store in the buffer the addresses of both LBL $f(x)$ and SIRTN itself, thus *there's no need to search for them in every iteration* of the solution, and we know there may be from several to many depending of the difficulty of the function. Consider that the OS routine [ASRCH] is used to locate them both, and it's a sequential search: first RAM for LBL $f(x)$, then ROM – and there may be several plugged in.

You'll no doubt notice that in the new SandMath there are only three functions related to this: **FROOT**, (not so fruity :-), **FINTG**, and **FLOOP** (the fluppy one :-) – which sure correspond to SOLVE, INTEG, and SILOOP. But what about the whereabouts of SIRTN? No, it's not one of the section headers - already used for other purposes- , and nor is it in the secondary FAT (that'd be impossible to pull off) – fortunately this is one of the added pluses of going bank-switched: SIRTN is in the FAT of bank-3 , all by itself so it'll be found while [ASRCH] is called from the MCODE... all that extra work payed off and so we saved a precious FAT entry in the main FAT.

All in all, a stroke of genius - with all the ingredients of a work of art if you ask me. So I feel especially glad to finally have cracked this nut and include it in the SandMath, the yellow ribbon around the box.

Hope this dissertation wasn't too boring, and that you enjoy it at least as much as I did working on it, although I doubt that can be possible.

Edited: 21 June 2013, 3:47 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Computing Sunrise/Sunset with the HP 35s

Posted by [Justin Sloan \(Honolulu\)](#) on 29 Apr 2013, 12:30 a.m.

This is a port of a program originally written for the HP-67 calculator and published as part of HP's Standard Pac. It can calculate the approximate sunrise and sunset time in GMT given a date and any position on earth in lat/lon. One of the goals of the port was to eliminate the need to use multiple labels, to essentially use branching instead of subroutines. I would love to hear your thoughts or suggestion for improvement.

- Day = Register A
- Month = Register B
- Latitude = Register C
- Longitude = Register D

In addition, Register E is used during the program execution. Enter positive values for both the northing and easting.

As an example, to calculate the approximate Sunset on 14 June in Honolulu, Hawai'i, you would press the following keys: 14, STO A, 6, STO B, 21.30, STO C, 157.85, STO D, GTO S001, R/S, 1, R/S.

The answer should be 5.12, meaning 0512hrs (5:12am) GMT. This would be 1912hrs (7:12pm) local HST, since HST is -10 from GMT.

- S001 LBL S
- S002 FIX 2
- S003 CF 2
- S004 0=SUNRISE,1=SUNSET
- S005 x>0?
- S006 SF 2
- S007 30.3
- S008 RCL B
- S009 1
- S010 -
- S011 x
- S012 RCL A
- S013 +
- S014 1
- S015 -
- S016 0.998
- S017 x
- S018 STO E
- S019 87
- S020 +
- S021 COS
- S022 0.123

- S023 x
- S024 RCL E
- S025 ENTER
- S026 +
- S027 20
- S028 +
- S029 SIN
- S030 6
- S031 /
- S032 -
- S033 CHS
- S034 RCL E
- S035 10
- S036 +
- S037 COS
- S038 23.5
- S039 x
- S040 TAN
- S041 RCL C
- S042 HMS>
- S043 TAN
- S044 x
- S045 ACOS
- S046 FS? 2
- S047 GTO S064
- S048 CHS
- S049 RCL D
- S050 HMS>
- S051 +
- S052 15
- S053 /
- S054 +
- S055 12
- S056 +
- S057 ENTER
- S058 CLx
- S059 x>y?
- S060 24
- S061 +
- S062 >HMS
- S063 RTN
- S064 RCL D
- S065 HMS>
- S066 +
- S067 15
- S068 /
- S069 +
- S070 12
- S071 +
- S072 24
- S073 x>y?
- S074 GTO S078
- S075 -
- S076 >HMS

- S077 STOP
- S078 ENTER
- S079 -
- S080 GTO S075

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

HP-42S with Electroluminescent screen and simple I/O port

Posted by [Jose Poyan](#) on 26 Mar 2013, 4:45 p.m.

Summary

This project is about modifying an HP-42S to include an electroluminescent back lighted display which would look very similar to an LED display but in blue-green colour. The aim was to make a prototype that looks exactly the same as a normal HP-42S externally except for the display colour when off (black) and I even wanted to do a custom box and show it as a finished new model, let's call it HP-42EL or HP-42EX (electroluminescent-expandable). The calculator would have included a very primitive I/O port using the keyboard. That was when I had access to a full equipped workshop. But that has changed and I am pretty much sure that I will not have access to it for many years to come. Hence I wanted to share my ideas with the users in this forum, but it is a bit of a shame for me that I can not make the article I wanted originally. The prototype was never completed, but very fortunately I took a couple of (cell phone) pictures of the prototype display fired up (tested on an old HP17B), and I did the dimensions to pack a Li-Ion battery into the case for a very short running time, which is a proof that it would have worked. This article is based in those pictures. Hopefully someone can use them to make his own electroluminescent calculator if he feels motivated...and has a workshop :-)

Introduction

When I was young I wanted to replace my aged and only calculator, a VFD vacuum fluorescent display Casio FX-110 with that horrible keyboard. I fixed in my mind that I wanted the same display but a more powerful machine, and RPN. But I could not find it nowhere. At that time I simply could not understand why no company would attempt to combine the excellent VFD display with a good set of functions. I finally bought an HP32SII but I always wanted THAT imaginary calculator or one with LED display since I had access to an HP25C...so I thought I could make one. I know a lot about electronics, in a scale of 1 to 10, I give myself a...0. Since I am a total ignorant in electronics I concentrated in another thing... the mechanical aspect of the construction. And I found there were opportunities with Pioneer model. I first took the HP-20S.

HP 20S with electroluminescent display

The original LCD display had its back film removed. A paper thin electroluminescent film was put instead. There was just about the right space to slide it from one side, in between the display glass and the mother board. (text describes the forward picture) The top was cut off from an almost destroyed HP17B. The top is removable and the calculator will run on its own batteries like a normal HP20S.



It does not look the same. The screen is a bit darker because the silver back film is not there anymore to reflect the light.



Turned On to make reading easier. It is more blue light than turquoise.





This is a more realistic view. The backlight is not so bright:



A power generator for one single 1.5Volt cell was used, bought online with the electroluminescent film. The circuit was packed and glued inside the box.



The battery fits just fine.



The power unit is fully removable. The battery cover can be put back on the calculator. The couple of contacts are for power. The single contact is there to keep it firm.







It was not firm enough so two screws needed to be installed.





Slightly out, showing the contacts.



Conclusion

The power unit does not generate enough power to give a pleasant glow in the display. The top removable box feels fragile and it would not be a good idea to carry the calculator in a pocket. It took a lot of work but at least this project was finished.

HP 42S with Electroluminescent Display and primitive I/O port.

This is the HP17B used for the project. This machine, bought online, was the most abused calculator ever seen by this user. Someone had hit it badly in its corners against the floor and it had pray open marks all around the case made with a big screw driver. At some point the tool found its way in between the shells and scratched deeply the front of the display, but miraculously it did not brake it. It was this what motivated the writer to experiment with this unit. Nevertheless the keyboard and screen were still working fine in this US model ! I love HP vintage calculators.

The top was cut to make the HP20S. The original back film of the LCD was removed. A new polarized film was placed but without the silver back. A lot of tests were done to find the best angle of polarization so the light pass through the digits with maximum bright when the user is sitting normally in front (case: myself). The idea was to emulate the LED display so the polarization was inverted. The electroluminescent film is powered here by a electroluminescent power source with 6AA batteries with give more than 100V AC on the left cables. The upper cables feed the calculator. The calculator shells just snap together allowing for testing the life display every time.



This is the picture this article is all about ! Several electro colours were tested. Originally red. But red turned out to be pink and not real LED pure red ! The same happened with turquoise which was the preferred colour for the writher emulating the Casio VFD. But it was actually light blue. Finally this green was found to be the best colour. Its really pleasant to see. It is slightly overdriven as it will be explained which makes it to shine really well.

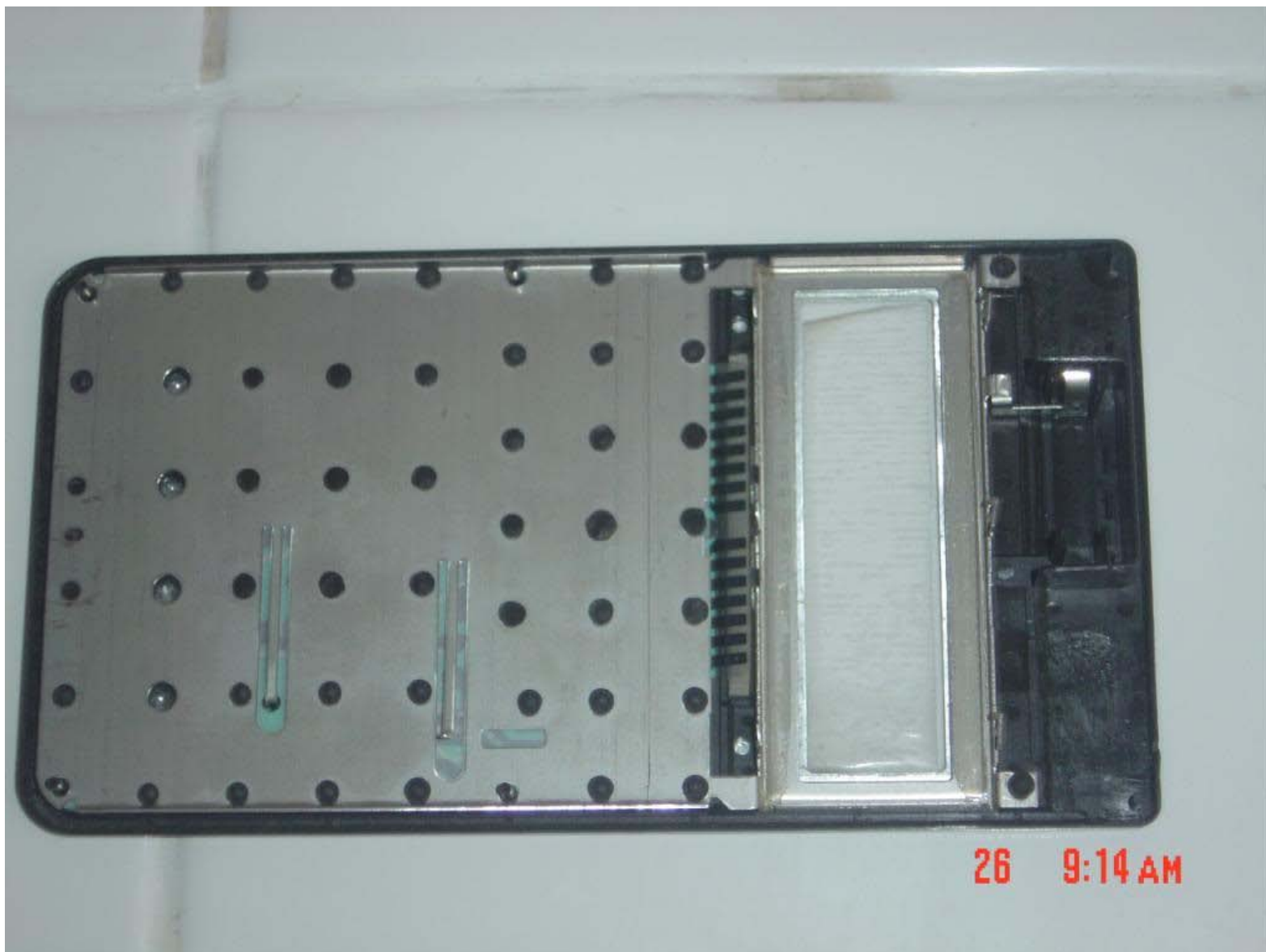


A second picture. The power supply was for testing. The prototype would have had a very small power source for two cells. Now that gives 3V. An this would be run with a Li-Ion Nokia 3.7 V BL-4C mobile battery. Over running the display would give the same bright on tests as this picture shows here ! (I never liked green a lot, but I loved this green, it is hypnotic, nice, like a red LED)



The first idea to power the display was to use micro Li-Pol cells that would cover all and every free space in between the lower shell and the mother board and would look as the carpet bombs on the skin of a Russian tank. Many sketches were done, concluding that it was possible to put up to 10 of them. But for the sake of simplicity the plan changed to one single Nokia cell. This had exactly the dimensions to be placed just on above the chip. But it would be needed to remove the central hook for the LCD and to cut the lower shell. It was necessary to make a tilted table to cut it, see later.

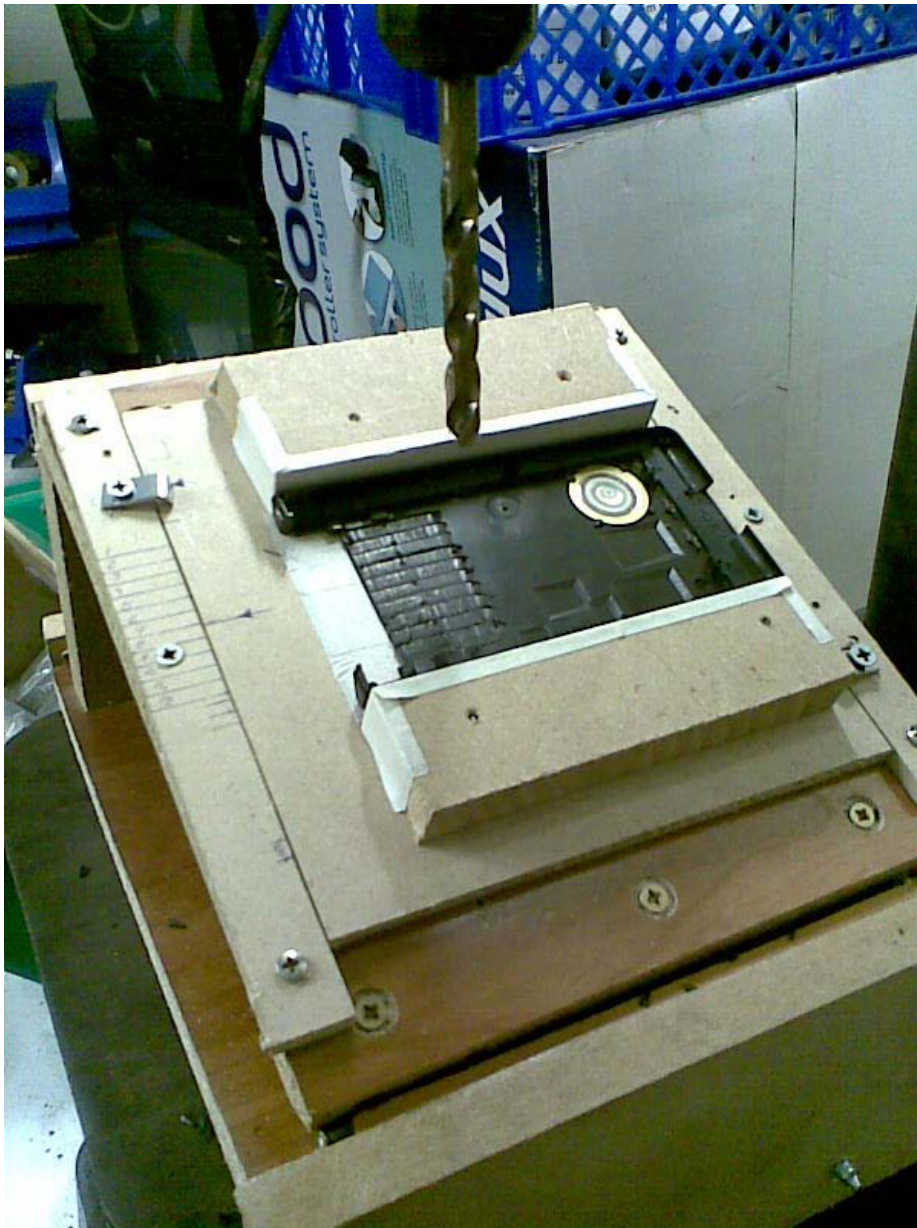
This picture shows the upper part emptied to allocate the power generator. Although there are not photos here, really a lot of work was done on the generator to be able to fit in this tight space. It was actually cut it in two pieces. There was one, and only one model of generator suitable for this project on sale online



The picture shows the hole to fit an iPod connector. This would give accessed to the keyboard and external power as well ! Only the iPod connector had enough pins to do this. It took five days of spare time to the writer to cut the lower hole with a Stanley knife to fit perfectly and nicely the plug.



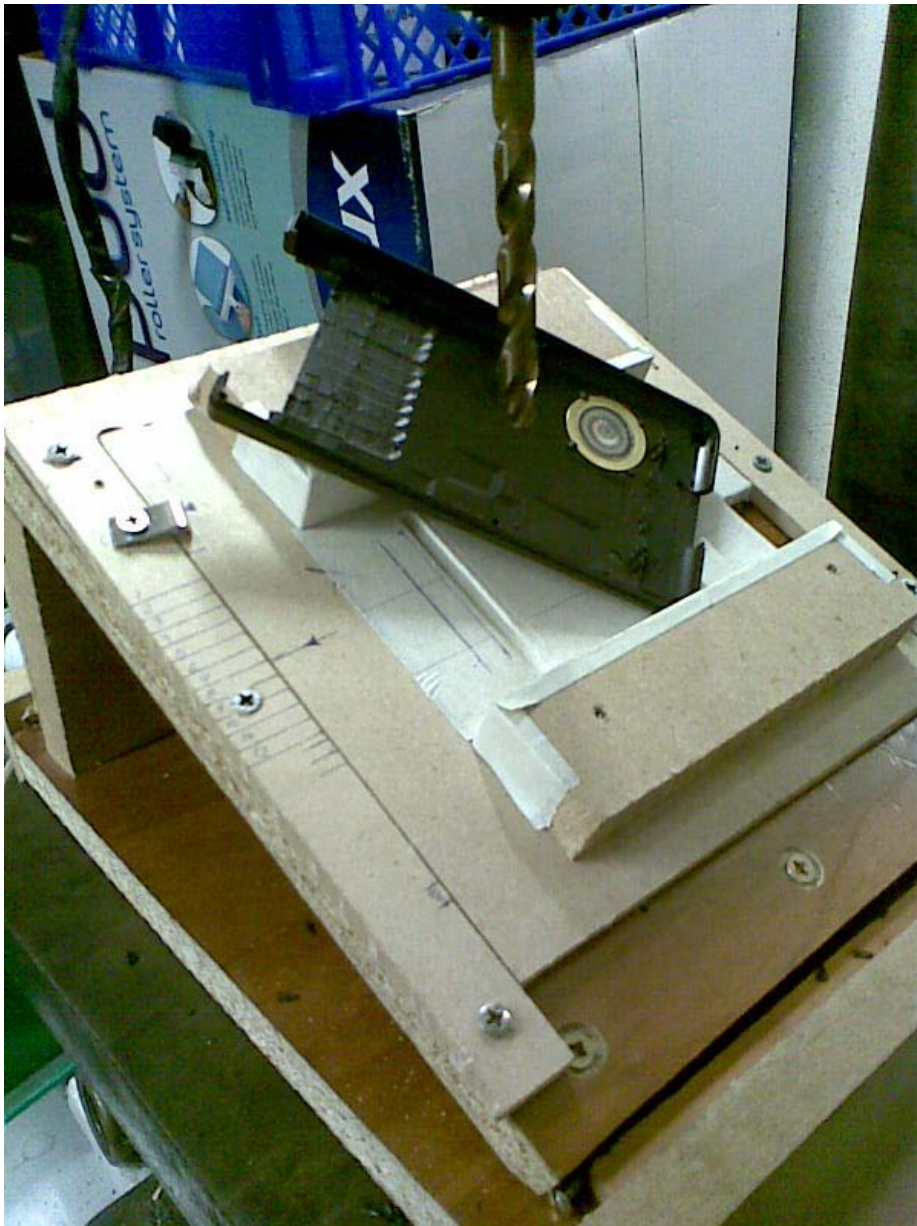
The d.i.y. tilted table used to cut space for the battery. It cuts half of the width, that is 1mm. The shell lays then on the battery itself. (The calculator is powered by its own cells)



Carefully cutting of the shell took one hour and two people even though the drill bit was new (sharp) it was not easy to cut.



Later on the surface was flattened with the Stanley knife.



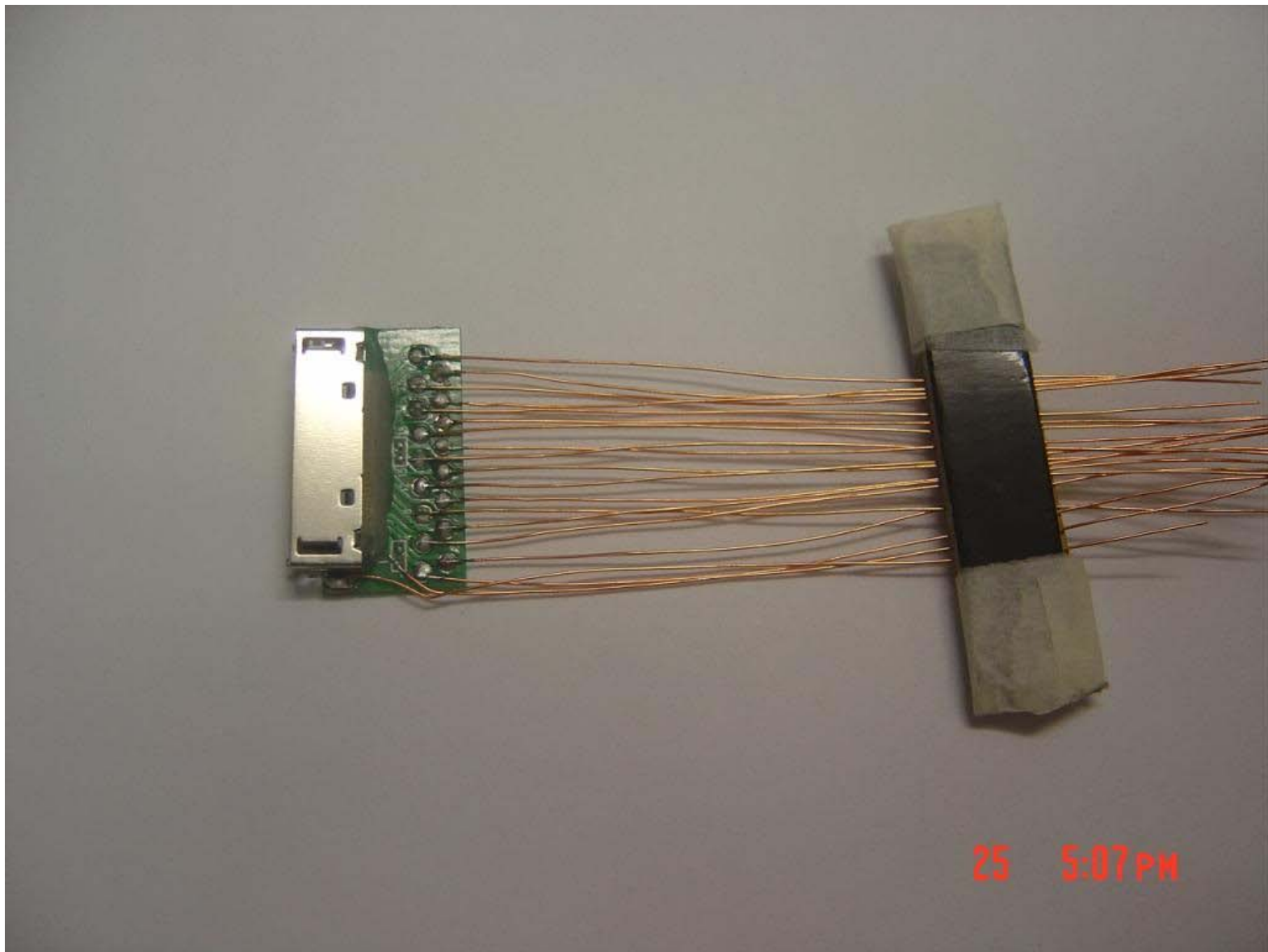
Nokia BL-4C. This battery is only 2mm thick:



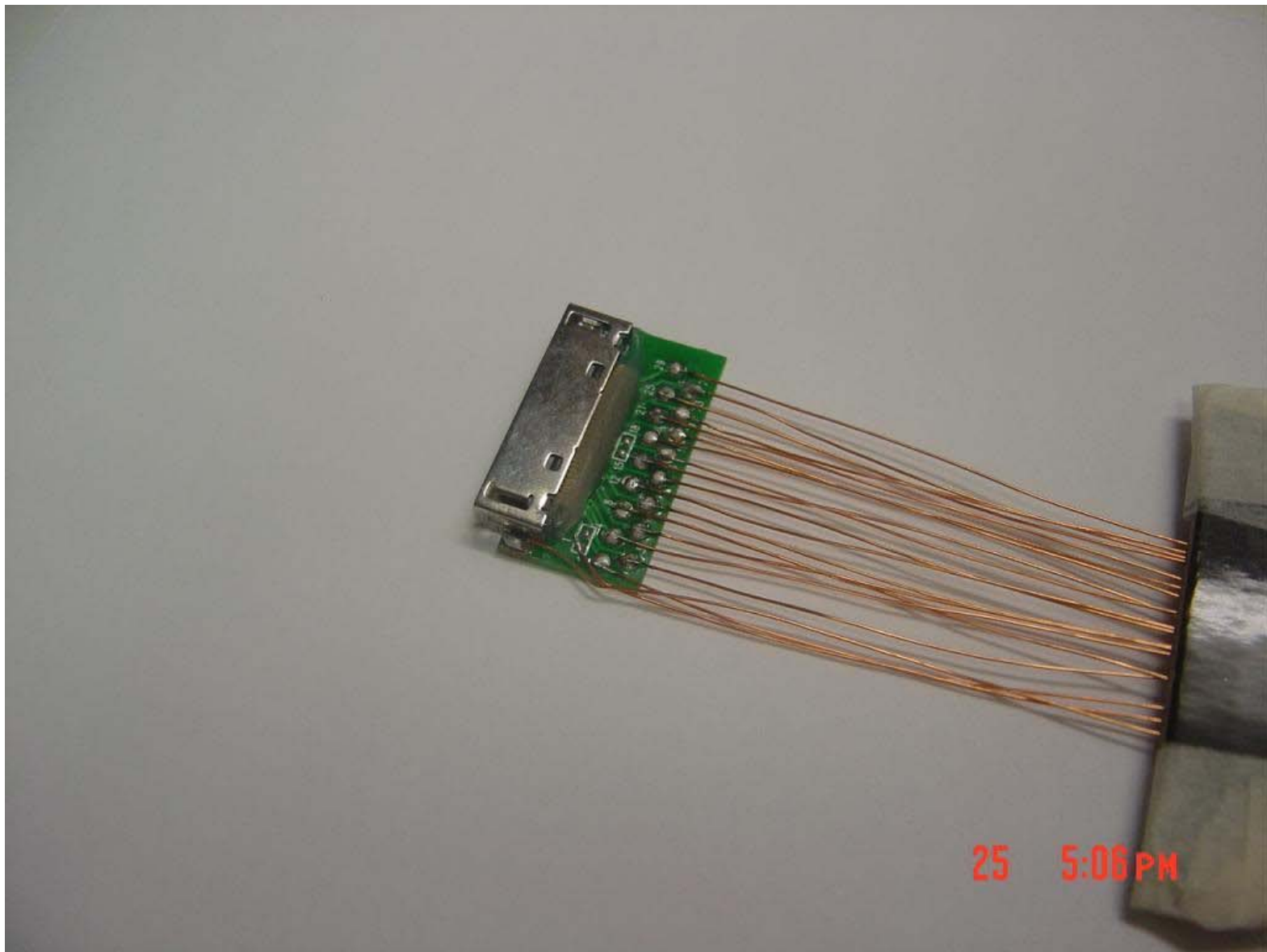
The core of the I/O port is this very small flexible printed circuit. It slides in between the mother board and the keyboard contacts. You can see the silver painted contacts on top. This was made with a plastic clear sheet and a copper tape, then cutting slowly the copper to get the strips. The lower end has a very thin piece of wood to separate the copper from the plastic and be able to solder the contacts to the iPod connector. Everything is painted in clear varnish. The connector on the right is the male iPod. A second prototype of the flexible board was built. The left connector shows a very rough iPod plug to be tested on the HP17B, but not on the HP42EL (I wanted to make a removable memory chip on the HP17B with a small back up battery. This connector was very thin and with a lot of contacts for the chip, but that was another project)



This is the female connector. If you turn it upside down you can image it resting on the shell you have seen above. It fits perfectly.



With some patient it was possible to use all plus one contact normally not used.



The project stopped here. The original Nokia 890mAh BL-4C battery gave a mediocre 1 hour runtime. I planned to make a docking station to run the power generator with an external battery. This would have given indefinitely running time really. The mayor problem was space and there was enough space to put a micro Nokia DC plug on the top. Alternatively the iPhone plug would provide power without the docking station but with a cable. The I/O port would have used a circuit interface on the form of a box that just writes on the keyboard at high speed using the iPod connector and some programming code. Transferring data to the computer it is another story. This thing was listed as “transfer all your data to your computer. It uses DOS.....” It would be possible to use something like this, I only saw it once on sale, but I least it exist:



Conclusion

The original design had for objective to show that is was possible to pack an electroluminescent display and power source into the sleek design of a Pioneer. The I/O port was not included then, later on I had the idea of the flexible printed board and I thought it was easy to implement. So removing that from the head and with the exception of the black display we can say that there would have not been physical difference from this calculator and a normal one. It would have looked exactly the same ! Even the connector for the external AC in (the docking station) would have been hidden behind the battery port. However, the plastic shells could be taken apart by hand, and to give the case strength it would have been necessary to put four small screws. That would have ruined the design, and moved out from the original idea. But in fact it would have been unnecessary because the plastic shells fit tight one against each other as you know if you have taken apart one hp42s and put back the shells together. Anyway I don't need the screws...at the end it is a prototype! I do like prototypes.

Jose Poyan

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

GTO

[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous](#) | [Next \]](#)

Sudoku Solver for the HP-15c

Posted by **Marcel Samek** on 4 Feb 2013, 1:20 p.m.

This program uses a backtracking algorithm to find the first solution to a sudoku puzzle.

If you wish to try it, you have to save the starting puzzle in registers 8-16. Each row of the sudoku is represented as a single, 9 digit number, with 0 representing each blank. Once you have done that, you can run the program with a GSB A. When it is complete, you will find the solution to the sudoku puzzle in registers 17 to 25. Unfortunately, you will need to use the indirect register to get at the result, because there is no way to directly see the value of register 20 and above. I am working on making the program a little smaller so that I can put in a loop to print the result out line by line.

It is quite slow, depending on the complexity of the puzzle. The complexity of the puzzle (as far as this program is concerned) depends on the number of clues, and their placement in the puzzle. The more clues, the faster it will solve. The more clues towards the upper part of the puzzle (especially the upper left), the faster it will solve. Some puzzles can take many hours to solve, and some just a few minutes. I have only tried it on the 15C LE which is about 100 times faster than the original 15C.

If you want to make sure that it works after you have typed it in, enter a solved sudoku with just a few values missing and see if it finds them correctly. It should be able to do that in a few seconds and it acts as a quick sanity check to make sure you keyed the program in correctly.

This is a work in progress and I am still trying to squeeze it down to make it more user friendly. The latest version will always be found at: [Sudoku Solver for the HP-15c](#)

```

Sudoku Solver for HP 15C
Special Thanks: To the MoHPC community for their many suggestions

Before entering the program perform the following:
34 DIM (i)

////////////////////////////////////
Register And Flag Usage
////////////////////////////////////
0      General purpose variable used for miscellaneous purposes
1      Current index (0-80) in the pseudo-recursion
2      Row (0-8) of current index
3      Column (0-8) of current index
4      Block # (0-8) of current index
5      Power of 10 of current column index
6      Value in the test solution at current index
7      Value of start clue at current index (0 if not set)
8 - 16 Starting row data
17 - 25 Current test solution
26 - 34 Flag matrix (bit set if digit used in a row/column/block)

Flag 2  Indicates that a digit has been used in cur row/column/block
Flag 3  Input to Subroutine B (whether to set or clear flags)

```

```
; setU(x)
; Set/clear flag matrix values (show that x is used in a row/column/block)
;
LBL D
    GSB 5      ; calc bit value we need to set/clear in existing row
    RCL 2      ; Get the current row index into x
    GSB B      ; set flag matrix value and calc new bit value for the column
    RCL 3      ; Get the current column index into x
    GSB B      ; set flag matrix values and calc new bit value for the block
    RCL 4      ; Get the current block index into x

; MUST IMMEDIATELY FOLLOW PRECEEDING SUBROUTINE
; utility subroutine for setting flag matrix values
```

```
LBL B
    GSB 1      ; get the current flag matrix row at index x

    RCL 0      ; get temp register (holds the bit value we will be setting)
    F? 3       ; flag 3 indicates if we are setting or clearing the flag
    CHS       ; if we are clearing, we will do a subtraction instead
    +         ; set/clear the flag

    X<>Y      ; bring the row index back into x
    2         ; 26 is the starting register for the flag matrix
    6
    GSB 3      ; set I so that we are ready to store the new value
    STO (i)    ; store the new value into the flag matrix
    RDN       ; get rid of the new value to restore the stack
    9         ; the next bit value will be 9 bits to the left
    +         ; set the next bit index
    GTO 5      ; calculate the value with that bit set
              ; we GTO instead of GSB and it will do the RTN
```

```
; putA(x)
; Set the value x into the current row/column in the trial solution.
; Does it by subtracting the previous value and adding the new one.
;
LBL 7
    X<>6      ; swap new value with register that holds current value
    STO 0      ; store the old value in the temp register
    RCL 2      ; Get the current row index into x
    1         ; 17 is the starting register for the current trial solution
    7
    GSB 3      ; Set the indirect register
    RCL (i)    ; Get the current value for the entire row
    RCL 6      ; Get the new value
    RCL- 0     ; subtract the old value from the new value
    RCL* 5     ; shift the power of 10 to the appropriate column
    +         ; add to the old value
    STO (i)    ; store the new row value from where we got it
    RTN
```

```
; change(x)
; Increments or decrements the current position in the trial solution.
; Updates the registers containing the current row, column and block index,
; and the one with the power of 10 factor for the current column and others
;
LBL 6
    STO+ 1     ; x holds +1 or -1; Register 1 is the current index

    RCL 1      ; get the current index (0 to 80)
    RCL 1      ; get the current index (0 to 80)
    9         ; integer divide by 9 to get the row index (0 to 8)
    /         ; no integer divide on 15c so do a floating point divide
    INT       ; use the INT operator to finish of the integer divide
    STO 2      ; register 2 contains the current row index

    9
    *
    -         ; col = index - 9 * row
```

```

STO 3      ; register 3 contains the current column index

3          ; calculate the block index from the row & column indexes
/          ; TODO: save a couple of bytes in this section of code
RCL 2
3
/
INT
3
*
+
STO 4      ; register 4 holds the block index

8          ; now calculate the power of 10 of the current column
RCL - 3    ; Get the digit (from right) based on the column
10^X      ; calculate the exponent
STO 5      ; save in register 5 which is used throughout the code

RCL 2      ; get the current row
1          ; 17 is the start register of the current trial solution
7
GSB 4      ; extract the value at the current column
STO 6      ; reg 6: the current trial value at the current row/column

RCL 2      ; get the current row
8          ; 8 is the start register of the input data from the user
GSB 4      ; extract the value at the current column
STO 7      ; reg 7: starting value at the current row/column (0 if none)
RTN

; Extract value at the current column from the matrix indirectly specified by x&y
LBL 4
GSB 3      ; set the indirect register based on x & y
RCL (i)    ; get the row from the matrix passed in
RCL / 5    ; shift the row to the right
INT        ; trim off the digits shifted to the right of the decimal
1          ; we will do a modulus 10 to extract the last digit
0
/          ; do the equivalent of a mod 10
FRAC
1
0
*
RTN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; main()
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LBL A
CF 2        ; make sure flag 2 is unset - CLR REG does not do this
CF 3        ; make sure flag 3 is unset - CLR REG does not do this
1          ; start with a index in register 1 of -1 (0 to 80)
CHS        ; that way we can start with an increment operation
STO 1       ; and actually start at 0 where we want.

LBL 2
; set the flags to show the input values are set
1          ; go forward one position at a time
GSB 6      ; go to the next position in the trial solution

RCL 7      ; get the starting input value at this row/col
GSB 7      ; set the value in the trial solution
RCL 7      ; get starting input value because the last call destroyed it
TEST 1     ; if > 0 then the user input a value for this row/col
GSB D      ; set the flags to indicate this value is set

8          ; 80 is the upper bound of the indexes (9x9 = 80 = 0:80)
0
RCL 1      ; get the current index
TEST 6     ; if the current index hasn't reached 80
GTO 2      ; do the next value
1          ; reset the starting value

```

```

CHS      ; to -1 as we did at the beginning of the program
STO 1    ; register 1 holds the current index

LBL E    ; main solution loop
8        ; when we reach the last index (80) we are done
0
RCL 1    ; register 1 holds the current index
TEST 5   ; see if we are at the end
RTN      ; finished ; woohoo - we are done!
1        ; Go forward one spot
GSB 6    ; Do the position increment
RCL 7    ; get the starting input value at this row/col
TEST 1   ; if it's > 0, the user specified a value here
GTO E    ; go forward, since this value was specified by the user
GSB 7    ; Set the value in the trial solution

LBL 8    ; check the possible digits in order 1-9.
9        ; Get the current trial solution value
RCL 6    ; Check to see if it is 9
TEST 5   ; If it is, backup one step
GTO C    ; We weren't at 9 yet, so increment the value by 1
1
+
GSB 7    ; Set the value in the trial solution

RCL 6    ; Get the current trial solution value
GSB 5    ; Calc 2^x-1 to get the bit mask
CF 2     ; Clear the flag thats used as a return value
RCL 2    ; Get the current row index into x
GSB 9    ; see if the current value has already been used in the row
F? 2    ; If number has been used in the block, try the next value
GTO 8
RCL 3    ; Get the current column index into x
GSB 9    ; see if current value has already been used in the column
F? 2    ; If number has been used in the block, try the next value
GTO 8
RCL 4    ; Get the current block index into x
GSB 9    ; see if the current value has already been used in the block
F? 2    ; If number has been used in the block, try the next value
GTO 8
RCL 6    ; Get the current trial solution value
GSB D    ; set the flags to indicate this value is set
GTO E    ; move on to the next position in the puzzle

LBL C    ; Come here to back up to the previous position
1        ; We will go one spot backwards
CHS
GSB 6    ; Set the new current position and all temp values
TEST 1   ; previous call leaves the starting value in X
GTO C    ; if value is > 0, it was set, backup one more spot
RCL 6    ; Get the current trial solution value

SF 3     ; flag 3: clear the flag matrix bits, instead of setting them
GSB D    ; Set/Clear the flag matrix bits
CF 3     ; unset the 3 flag
GTO 8    ; check the next digit

LBL 9    ; get the appropriate row (x) from the flag matrix
GSB 1    ; divide by the temp register - right shifts value
RCL / 0
INT
2        ; if bit is set, fractional part will be non 0 when / 2
/
FRAC
TEST 1   ; if bit is set, set flag 2 which is used as a return value
SF 2
RDN      ; move the stack down to prepare the caller for the next call
RDN      ; move the stack down to prepare the caller for the next call
9        ; bit flags for row/col/block are << by 9 from each other
+        ; calculates the appropriate bit offset for the next call
GTO 5    ; calc 2^x-1 to get the bit mask
        ; do a GTO instead of GSB and it will return for us

```



```

////////////////////////////////////
; setPow2(x)
; Sets the utility temp register to 2^(x-1). Leaves x in place.
////////////////////////////////////
LBL 5
    STO 0      ; store the input X in the temp register
    1          ; we want to subtract 1 from the exponent
    -          ; calculate x-1
    2          ; set the base as 2
    X<>Y       ; the y^x function wants x and y reversed
    y^x        ; calculate the value
    X<>0       ; stuff result in temp register and restore the input x
    RTN

////////////////////////////////////
; getPart(x)
; Returns the integer representing the entire Xth row of the flag matrix
; Row numbers start at 0.
; returns value in x - input parameter x ends up in y
////////////////////////////////////
LBL 1
    ENTER
    ENTER      ; duplicate the parameter so we can leave it for the caller
    2          ; 26 is the starting register for the flag matrix
    6
    GSB 3      ; set the indirect register to the row specified by x
    RCL (i)    ; retrieve the entire row from the flag matrix
    RTN

; Set the indirect register and remove the parameters from the stack
LBL 3
    +          ; x+y is the memory offset we want
    STO I      ; put it in the indirect register
    RDN        ; get rid of the sum from the stack
    RTN

```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Mapping HP48 Text to Unicode

Posted by [Chris Dreher](#) on 16 Jan 2013, 1:15 a.m.

Here is a table for how to translate HP48 text characters into Unicode characters, which is what modern computers use for text. Now HP48 developers create software that transfers, displays, or edits HP48 characters (ex: copying a file from an HP48 calc to a computer) with code that will reliably display a corresponding character. By using this mapping table, we should be able to avoid the garbage data, bugs, and crashes that has been an issue for some PC/Mac/Linux side HP48 software.

Further details and explanations in an easier to read format is available at <http://www.drehersoft.com/mapping-hp48-text-to-unicode>

Mapping HP48 Text to Unicode

Most HP48 characters can be directly mapped to Unicode characters for the ranges of 0x00 to 0x1E, 0x20 to 0x7E, and 0xA0 to 0xFF. For example, an HP48 character of 'A' is 0x41 (65 in decimal) and in Unicode would be 0041 (65 in decimal). However, the 34 special characters of 0x1F and from 0x7F to 0x9F should be translated by the below table:

HP48		Unicode	
Dec	Hex	Code	Name
31	1F	2026	Ellipsis
127	7F	2592	Medium Shade
128	80	2220	Measured Angle
129	81	0101	Latin Small Letter a with Macron
130	82	2207	Nabla
131	83	221A	Square Root
132	84	222B	Integral
133	85	03A3	Greek Capital Letter Sigma
134	86	25B6	Black Right-Pointing Triangle
135	87	03C0	Greek Small Letter Pi
136	88	2202	Partial Differential
137	89	2264	Less-Than or Equal To
138	8A	2265	Greater-Than or Equal To
139	8B	2260	Not Equal To
140	8C	03B1	Greek Small Letter Alpha
141	8D	2192	Rightwards Arrow
142	8E	2190	Leftwards Arrow
143	8F	2193	Downwards Arrow
144	90	2191	Upwards Arrow
145	91	03B3	Greek Small Letter Gamma
146	92	03B4	Greek Small Letter Delta
147	93	03B5	Greek Small Letter Epsilon
148	94	03B7	Greek Small Letter Eta
149	95	03B8	Greek Small Letter Theta
150	96	03BB	Greek Small Letter Lamda
151	97	03C1	Greek Small Letter Rho
152	98	03C3	Greek Small Letter Sigma
153	99	03C4	Greek Small Letter Tau
154	9A	03C9	Greek Small Letter Omega
155	9B	0394	Greek Capital Letter Delta
156	9C	03A0	Greek Capital Letter Pi
157	9D	03A9	Greek Capital Letter Omega
158	9E	25A0	Black Square
159	9F	221E	Infinity

If you are using UTF-8, then it is necessary to encode each Unicode characters into the appropriate 1, 2, or 3 byte sequences.

[bold]Rationale[/bold]

In some cases, the choice of what Unicode character to use was trivial. Some of the below were not.

1. Character 0x80 (angle)

1. Instead using 2220 for character 0x80, others have incorrectly used 221F. 221F is the Right Angle character and is not intended for any generic angle. Also, it does not visually match the HP48.
2. While 2221 is visually an even better match, this character often does not render properly on various computer platforms and software. In short, some users will just see empty boxes.

2. Character 0x81 (x-bar)

1. In theory, Unicode allows two characters to be visually combined if the 2nd character is a "combining character". This would allow for the display of an x with a "combining macron" character, which would be 0078 followed by 0304. However, there are two problems with this.
 1. This combining of these two characters often renders poorly or not at all and will leave the user confused.
For additional examples of how x-bar is inconsistently rendered based on font, go <http://www.kreativekorp.com/charset/encoding.php?file=hp-48.kte&char=81>.
 2. Using two characters to represent one HP48 character breaks the pattern having a simple one-to-one mapping. Some HP48 developers will likely have bugs in the code when converting back from Unicode to HP48 characters.
2. Instead, a-bar (0101) is used. It is a single Unicode character so it is easy for HP48 developers to deal with, leading to less bugs. Also, x-bar is used in statistics as the notation for average. The 'a' in a-bar looks like an 'a' for average.

3. Character 0x82 (nabla)

1. The character 2207 was chosen over other triangles since this is the Nabla character which is used in mathematics. Details can be read http://en.wikipedia.org/wiki/Nabla_symbol.

4. Characters 0x8D through 0x90 (arrows)

1. In Unicode, there are a large number of characters that represent arrows. However, 2190 through 2193 were chosen because these are just simple arrow characters and don't carry any additional implied meaning. Also, this set of arrow characters supports all four directions where as some of the other sets do not. Lastly, some of the alternative arrow characters do not consistently get rendered on some computing platforms.

5. Characters 0x85, 0x8C, 0x9B, 0x9C, 0x9D (various Greek symbols)

1. These are Greek symbols that could have alternatively been represented by various mathematical or electrical Unicode characters. However there are several reasons for preferring the Greek symbols:
 1. We can gain insight into the original HP48 developers intentions by looking at how they translated these characters when using ASCII transfer mode 2 or 3 over a serial link. These characters were translated into \GS, \Ga, \GD, \PI, and \GW respectively. If we assume that "G" stands for Greek, then we can assume these translations mean Greek Capital Sigma, Greek lower alpha, Greek Capital Delta, Capital Pi, and Greek Capital Omega (a lower omega looks like a 'w'). This pattern holds for all the other translated Greek letters as well, except for \pi which is trivial to see that this is lower pi.
 2. Using all Greek symbols results in a visually clean look. In contrast, when symbols from math, electronics, and Greek symbols are mixed together, they often look sloppy because they don't line up, have different line weights, and different drawing styles.

6. Character 0x9E (box)

1. Instead of using 25A0 as the Black Box, others have incorrectly used 25AC which is the Black Rectangle. This visually does not match.

Resources

Unicode Standard: <http://unicode.org/>

Unicode Character Name Index: <http://www.unicode.org/charts/charindex.html>

UTF-8 summary: <http://en.wikipedia.org/wiki/Utf-8>

Edited: 16 Jan 2013, 1:16 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

HP-41 "Halfnut": patching the ROM for full lower case character set display

Posted by *J-F Garnier* on 28 Dec 2012, 12:25 p.m.

This article describes a simple patch of the HP-41 OS ROM to give access of a full lower case character set on the "halfnut" LCD.

Warning: using these patches on a "fullnut" HP-41 would give unexpected results: the lower case 'a' to 'e' characters may not be displayed at all any more!

These patches are to be used on an emulated HP-41, either by software or hardware. There is no way to modify the internal OS ROM of a standard HP-41.

The latest version of the HP-41 hardware (the "halfnut" version) includes several hardware enhancements (see ref.1). One is the support of the full lower case character set on the LCD. Unfortunately, the system ROM have not been updated by HP at the time to manage these extra characters, resulting in that all lower case characters above 'e' are actually displayed as the "fullstar" character.

The internal routine which manages the LCD display is the MASK entry point (\$2C88). The purpose of this routine is to convert an ASCII character to LCD character form (not including comma, period and colon), see ref.2.

The patches are indicated below in **bold** .

```
=MASK
2C88 016      A=0      XS
2C89 130 020  LDI      020 (32)
2C8B 306      ?A<C    X      ascii < 32 ?
2C8C 097      GOC      2C9E  yes, special char
2C8D 130 05F  LDI      05F (95)      ***
2C8F 0A6      ACEX     X      ***
2C90 306      ?A<C    X      ascii > 95 ?
2C91 0DF      GOC      2CAC  yes, new char      ***
2C92 3D8      CSTE X
2C93 144      ST=0    6      mask 6 bits only
2C94 3D8      CSTE X
2C95 106      A=C     X
2C96 1B0      C=STK
2C97 330      CXISA
2C98 170      STK=C
2C99 0A6      ACEX     X
2C9A 346      ?A#0    X
2C9B 3A0      RTNNC
2C9C 006      A=0     X
2C9D 31B      GONC    2C80
2C9E 04E      C=0     W      check special char table
2C9F 15C      PT=     6
2CA0 090      LC      2      table entry at 2C00
2CA1 310      LC      C
2CA2 01C      PT=     3
2CA3 330      CXISA
2CA4 366      ?A#C    X      load 1 char from table
                                match a special char ?
```


2CA5	033	GONC	2CAB	yes
2CA6	222	C=C+1	PT	point to next word
2CA7	3E3	GONC	2CA3	go on !
2CA8	130 03A	LDI	03A (58)	all segment if not found
2CAA	35B	GONC	2C95	
2CAB	03C	RCR	3	
2CAC	056	C=0	XS	
2CAD	236	C=C+1	XS	C[2:0] has the special char
2CAE	323	GONC	2C92	replace it ***

A side effect of these patches is that characters above 127 are no more displayed as the "fullstar" character, they may be displayed as various characters and may even include the colon symbol. This affects only the LCD display, not the ALPHA register which still holds the correct characters.

References:

- 1. "Halfnut - An Internally Improved HP-41C,CV,CX", by Richard Nelson & Jeremy Smith, CHHU V2N4 p9 (July/August 1985).
- 2. Coconut mainframe VASM listing, HP NOMAS, 1985.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

41-CL Routine of the week: MMUSWAP

Posted by [Ángel Martín](#) on 4 Nov 2012, 5:57 a.m.

Swapping MMU Pages

Often I get in a situation with my CL setup where I need to exchange the location of two pages; the virtual equivalent to swapping two 4k-modules plugged in the "normal" 41 machine ports.

The short routine below is a practical solution using a couple of functions from the AMC_OS/X (can be replaced with the same functions in the CDD or standard OS/X Modules).

It works by exchanging the MMU mappings to the page pointers, as simple as that: no need to type the ROM ID's, or do any data copying - such is the power of the MMU design.

Simply enter the FROM:/TO: page numbers in Y and X respectively (in decimal), then execute MMUSWP and the job is done.

Valid inputs are 8 to 15. The CL will prevent you from re-mapping the OS, but stay clear form 0-5 in the inputs to avoid getting in trouble. Better yet, modify the program to prevent any accidental invalid page range.

Page 4 is definitely to be excluded. It's up to you to allow 6 and 7 as well, just remember the OS rules about the printer and HPIL pages.

See the code below, short and sweet as promised:

```

01      LBL "MMUSWP"
02      8
03      WSIZE
04      RDN
05      XEQ 00
06      AST0 T
07      XEQ 00
08      AST0 L
09      XEQ 02
10      R^
11      ST0 L
12      RDN
13      LBL 02
14      "804"
15      ARCLH
16      "|-0"
17      ARCL L
18      YPOKE
19      X<>Y
20      RTN
21      LBL 00
22      "804"
23      ARCLH
24      "|-0-1111"
25      YPEEK
26      ASHF

```

```
27      X<>Y
28      END
```

Note: first byte in lines 16 and 24 is the APPEND character, "|-"

Caveat Emptor:

First off, this is for single pages and not for complete modules. Using it on multiple-page modules will of course create havoc with their FATs and general arrangements.

Second, if including the AMC_OS/X itself in the swapping it should be in the FROM: location (Y register). Otherwise it'll get unmapped before the program ends, and won't be possible to use the remaining ARCLH instruction.

For the same reason swapping the page mapped to the YFNS will be a problem if it's first un-mapped, as the following YPOKE instruction will NOT be found by the OS. Moreover, even using it in the FROM: location won't suffice: the last YPOKE line will be found but it'll then attempt to "re-map YFNS from itself", which will freeze the calculator just the same. In summary, do NOT attempt to swap the YFNS page at all.

Better safe than sorry: insert this subroutine before line 21:

```
23      *LBL 01
24      "OS AREA"
25      6          can use 7/8 to protect
26      X>Y?      the HPIL and PRINTER
27      PROMPT
28      X>Y?
29      GTEND     go to .END.
30      RDN
31      "PG=YFNS"
32      YFNZ?
33      X=Y?
34      PROMPT
35      X=Y?
36      GTEND     go to .END.
37      RDN
38      X<>Y      swap FROM:/TO:
39      RTN
```

And insert these two lines before line 01 to "anchor" YFNS to its current page:-

```
01 XEQ 01
02 XEQ 01
```

./

Edited: 4 Nov 2012, 2:55 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Palmer's Matrix Multi-Tool Extension

Posted by [mbrethen](#) on 30 Sept 2012, 4:43 p.m.

In 2008, Palmer O. Hanson, Jr. posted an extension to Stefan Vorkoetter's Matrix Multi-Tool for the HP-35s. Used in conjunction with Stefan's program, it provides additional matrix functions that you would find on an HP-28s or TI-86. This article proposes the following changes to his existing program:

1. Adds a new option to call Stefan's program
2. Makes the Utility (MOV), Math, Identity Matrix and Transpose routines callable as subroutines from other programs
3. Removes vector multiplication (available in Stefan's Matrix Multi-Tool)
4. Shortens menus (where practical) to fit the display
5. Pressing R/S without selecting a choice will return you to the previous menu

Using the Program:

Start the program by pressing XEQ N ENTER. This will display the main menu:

```
1M 2U 3+ 405Ab
```

The menu choices are:

1. Norm (magnitude) computation menu for matrix A.
2. Utility functions menu.
3. Arithmetic functions menu.
4. Matrix operations menu.
5. Input/output using Matrix Multi-Tool.

Pressing R/S without selecting a choice exits the program.

Utility Functions

Press 2 R/S from the main menu to display the utility functions menu:

```
1>C 2C> 3XC 4>D 5D> 6XD
```

The menu choices are:

1. Store the contents of Matrix A in Matrix C.
2. Recall the contents of Matrix C to Matrix A.
3. Exchange the contents of Matrix C with the contents of Matrix A.
4. Store the contents of Matrix A to Matrix D.
5. Recall the contents of Matrix D to Matrix A.

6. Exchange the contents of Matrix D with the contents of Matrix A.

Matrix Math Functions

Press 3 R/S from the main menu to display the math functions menu:

1A*C 2A+C 3A-C

The menu choices are:

1. Multiply the contents of Matrix A times the contents of Matrix C.
2. Add the contents of Matrix A and Matrix C.
3. Subtract the contents of Matrix C from the contents of Matrix A.

Performing Matrix Operations

Press 4 R/S from the main menu to display the matrix operations menu:

1DP 2Id 3K 4A^t

The menu choices are:

1. Compute the Dot Product of Matrix A and Matrix C.
2. Replace the contents of Matrix A with an Identity Matrix.
3. Fill the elements of Matrix A with a constant value, K.
4. Perform the transpose of Matrix A.

Matrix Multi-Tool Input/Output

Press 5 R/S from the main menu to run the Matrix Multi-Tool program.

Program usage is discussed in more detail [here](#). Verification exercises are also provided.

Program Listing

```

N001      LBL  N
N002      CLX
N003      SF 10
N004      EQN 1M 2U 3+ 405Ab
N005      CF 10
N006      x=0?
N007      RTN
N008      1
N009      x=y?
N010      GT0 N028
N011      x<>y
N012      2
N013      x=y?
N014      GT0 N125
N015      x<>y
N016      3
N017      x=y?
N018      GT0 N212
N019      x<>y
N020      4
N021      x=y?
N022      GT0 N314
N023      x<>y
N024      5
N025      x=y?
N026      GT0 N419
N027      GT0 N001
N028      CLX
N029      SF 10
N030      EQN 1ABS 2COL 3ROW

```



```

N031      CF 10
N032      x=0?
N033      GT0 N001
N034      1
N035      x=y?
N036      GT0 N046
N037      x<>y
N038      2
N039      x=y?
N040      GT0 N063
N041      x<>y
N042      3
N043      x=y?
N044      GT0 N090
N045      GT0 N028
N046      RCL N
N047      x^2
N048      1
N049      -
N050      1E3
N051      ÷
N052      ST0 I
N053      0
N054      RCL (I)
N055      x^2
N056      +
N057      ISG I
N058      GT0 N054
N059      SQRTx
N060      ST0 A
N061      VIEW A
N062      GT0 N028
N063      RCL N
N064      1
N065      -
N066      1E3
N067      ÷
N068      ST0 I
N069      ST0 J
N070      CLx
N071      ENTER
N072      0
N073      RCL (I)
N074      ABS
N075      +
N076      ISG I
N077      GT0 N073
N078      x<y?
N079      x<>y
N080      RCL N
N081      1E3
N082      ÷
N083      ST0+ I
N084      x<>y
N085      ISG J
N086      GT0 N072
N087      ST0 C
N088      VIEW C
N089      GT0 N028
N090      RCL N
N091      ST0 I
N092      1
N093      -
N094      1E3
N095      ÷
N096      ST0x I
N097      ST0 J
N098      RCL N
N099      1E5
N100      ÷
N101      ST0+ I
N102      CLx
N103      ENTER
N104      0
N105      RCL (I)
N106      ABS
N107      +
N108      ISG I
N109      GT0 N105
N110      x<y?
N111      x<>y

```

```
N112      RCL N
N113      x^2
N114      1
N115      -
N116      ST0 - I
N117      1E -3
N118      ST0+ I
N119      REGZ
N120      ISG J
N121      GT0 N104
N122      ST0 R
N123      VIEW R
N124      GT0 N028
N125      CLx
N126      SF 10
N127      EQN 1>C    2C>    3XC  4>D    5D>    6XD
N128      CF 10
N129      x=0?
N130      GT0 N001
N131      1
N132      x?y?
N133      GT0 N136
N134      XEQ N165
N135      GT0 N125
N136      x<>y
N137      2
N138      x?y?
N139      GT0 N142
N140      XEQ N175
N141      GT0 N125
N142      x<>y
N143      3
N144      x?y?
N145      GT0 N148
N146      XEQ N185
N147      GT0 N125
N148      x<>y
N149      4
N150      x?y?
N151      GT0 N154
N152      XEQ N167
N153      GT0 N125
N154      x<>y
N155      5
N156      x?y?
N157      GT0 N160
N158      XEQ N177
N159      GT0 N125
N160      x<>y
N161      6
N162      x=y?
N163      XEQ N187
N164      GT0 N125
N165      XEQ N204
N166      GT0 N168
N167      XEQ N196
N168      RCL(I)
N169      ST0(J)
N170      1
N171      ST0+ J
N172      ISG I
N173      GT0 N168
N174      RTN
N175      XEQ N204
N176      GT0 N178
N177      XEQ N196
N178      RCL(J)
N179      ST0(I)
N180      1
N181      ST0+ J
N182      ISG I
N183      GT0 N178
N184      RTN
N185      XEQ N204
N186      GT0 N188
N187      XEQ N196
N188      RCL(I)
N189      x<>(J)
N190      ST0(I)
N191      1
N192      ST0+ J
```

N193	ISG	I		
N194	GT0	N188		
N195	RTN			
N196	RCL	N		
N197	ENTER			
N198	x^2			
N199	+			
N200	1			
N201	+			
N202	ST0	J		
N203	GT0	N206		
N204	0			
N205	ST0	J		
N206	RCL	N		
N207	x^2			
N208	ST0+	J		
N209	1			
N210	-			
N211	1E3			
N212	÷			
N213	ST0	I		
N214	RTN			
N215	CLX			
N216	SF	10		
N217	EQN	1A*C	2A+C	3A-C
N218	CF	10		
N219	x=0?			
N220	GT0	N001		
N221	1			
N222	x?y?			
N223	GT0	N227		
N224	XEQ	N238		
N225	XEQ	N185		
N226	GT0	N215		
N227	x<>y			
N228	2			
N229	x?y?			
N230	GT0	N233		
N231	XEQ	N287		
N232	GT0	N215		
N233	x<>y			
N234	3			
N235	x=y?			
N236	XEQ	N289		
N237	GT0	N215		
N238	RCL	N		
N239	ENTER			
N240	x^2			
N241	*			
N242	4			
N243	÷			
N244	2			
N245	+			
N246	INTG			
N247	ST0	T		
N248	VIEW	T		
N249	RCL	N		
N250	ENTER			
N251	x^2			
N252	2			
N253	*			
N254	+			
N255	ST0	E		
N256	RCL	N		
N257	1			
N258	-			
N259	1E3			
N260	÷			
N261	ST0	V		
N262	RCL	N		
N263	ST0	J		
N264	x^2			
N265	ST0+	J		
N266	ST0+	J		
N267	ST0	I		
N268	ST0+	I		
N269	1			
N270	ST0-	J		
N271	ST0-	I		
N272	-			
N273	1E3			

```

N274      ÷
N275      ST0+ I
N276      RCL (I)
N277      ST0 (J)
N278      DSE J
N279      DSE I
N280      GT0 N276
N281      XEQ M403
N282      ISG V
N283      GT0 N262
N284      CF 0
N285      CF 2
N286      RTN
N287      1
N288      GT0 N290
N289      -1
N290      ST0 S
N291      RCL N
N292      x^2
N293      ST0 J
N294      1
N295      -
N296      1E3
N297      ÷
N298      ST0 I
N299      RCL (I)
N300      RCL (J)
N301      RCL S
N302      *
N303      +
N304      ST0 (J)
N305      1
N306      ST0+ J
N307      ISG I
N308      GT0 N299
N309      XEQ N185
N310      RTN
N311      CL x
N312      SF 10
N313      EQN 1DP 2Id 3K 4At
N314      CF 10
N315      x=0?
N316      GT0 N001
N317      1
N318      x=y?
N319      GT0 N337
N320      x<>y
N321      2
N322      x?y?
N323      GT0 N327
N324      XEQ N361
N325      XEQ N185
N326      GT0 N311
N327      x<>y
N328      3
N329      x=y?
N330      GT0 N376
N331      x<>y
N332      4
N333      x=y?
N334      XEQ N388
N335      XEQ N185
N336      GT0 N311
N337      RCL N
N338      x^2
N339      ST0 J
N340      1
N341      -
N342      1E3
N343      ÷
N344      ST0 I
N345      0
N346      RCL (I)
N347      RCLx (J)
N348      +
N349      1
N350      ST0+ J
N351      Rdn
N352      ISG I
N353      GT0 N346
N354      ENTER

```

```
N355      SF 10
N356      EQN DP =
N357      PSE
N358      CF 10
N359      STOP
N360      GT0 N311
N361      XEQ M199
N362      CLX
N363      DSE J
N364      STO (J)
N365      DSE I
N366      GT0 N363
N367      1
N368      STO (J)
N369      RCL+ N
N370      STO+ J
N371      RCL J
N372      RCL E
N373      x>y?
N374      GT0 N367
N375      RTN
N376      INPUT K
N377      RCL N
N378      x^2
N379      STO I
N380      1
N381      STO- I
N382      RCL K
N383      STO (I)
N384      RCL I
N385      x?0?
N386      GT0 N380
N387      GT0 N311
N388      RCL N
N389      x^2
N390      STO W
N391      STO J
N392      1
N393      -
N394      1E3
N395      ÷
N396      STO I
N397      RCL (I)
N398      STO (J)
N399      RCL N
N400      STO+ J
N401      x^2
N402      2
N403      *
N404      RCL J
N405      x<y?
N406      GT0 N411
N407      1
N408      STO+ W
N409      RCL W
N410      STO J
N411      ISG I
N412      GT0 N397
N413      RTN
N414      CF 0
N415      XEQ M001
N416      GT0 N001
```

Length = 1428; Checksum = 9CDF

Note: The keystrokes to enter the superscript "t" in line N313 are: (yellow left shift key) CONST "atm" ENTER
BACKSPACE LEFT-CURSOR BACKSPACE ENTER

Revision History

2012 Sep 30 — Initial release.

2012 OCT 02 — Made the Utility (MOV), Math, Identity Matrix and Transpose routines callable.

2012 OCT 09 — Removed the matrix-vector product Ab function. This is available in the main menu of the Matrix Multi-Tool program, which is more convenient for viewing results.

2012 OCT 29 — No longer use flag to switch between matrix store/recall.

Edited: 29 Oct 2012, 6:22 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-UX and RPL/2

Posted by **[BERTRAND Joël](#)** on 26 Sept 2012, 6:03 a.m.

Hello,

I have seen that someone has trouble to build RPL/2 on HP-UX. RPL/2 doesn't contain any linuxism and runs fine on several Unix without any modifications. I have tested : Solaris, old HP-UX release, Linux, Net/Free/OpenBSD and some others. Thus, you can build RPL/2 on HP-UX with gcc/g++/gfortran toolchain and _g_make (BSD make is not enough to completely build RPL/2).

If you have some trouble, please ask to mailing list (<https://www.systella.fr/cgi-bin/mailman/listinfo/rpl2>).

Regards,

JKB

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

41-MCODE: Breaking the FAT Barrier

Posted by [Ángel Martín](#) on 3 Sept 2012, 6:16 a.m.

Breaking the 64-Entries FAT barrier: Multi-function groups.

The following article describes the most important considerations about the Multi-Functions implementation, highlighting their advantages and few shortcomings. The author has used the excellent HEPAX implementation as starting basis, adding functionality to the original design to improve usability and ease of use – while adapting it to the POWERCL structure.

Definition: A multi-function group consists of a set of sub-functions not included in the main FAT, but that nevertheless can be called individually using dedicated launchers by their own name or index within the group. Furthermore, they can also be used in a program using the same method – thus those launchers shall be “programmable” and have the necessary logic to invoke the appropriate sub-function upon request.

Examples in the HEPAX are the XF/XFA group and the HEPAX/HEPAXA group.

Benefits of the Multi-Function groups.

1. Unlimited number of sub-functions within each group.
2. Sub-functions can be grouped in separate FATs
3. Sub-functions can be called by Name and by index
4. They are all programmable, using the non-merged scheme.

Limitations of the original HEPAX implementation:

1. Two main functions (per each Multi-Function group) are required in order to access them by name and by index. On the HEPAX the examples were XF/XFA, as well as HEPAX/HEPAXA.
2. No CATALOG was available to list/enumerate the sub-functions included under each group.
3. Individual sub-functions can't be assigned to a key.
4. Sub-functions can't be prompting functions.
5. Since they're usually allocated on secondary banks, they can't use the Partial Data Entry technique (light sleep re-activates bank-1).

Naturally the goal in was to overcome as much as possible many if not all of the limitations, which with the exception of them being assignable has been accomplished – plus a couple of extras have been thrown in to round up the functionality:-

Improving on such a masterpiece has been possible thanks to the initial implementation being so well structured and designed: wonderful programming to begin with!

With a little effort and more available space (thanks to the Library#4 assistance!), here are the main changes and additions made.  

Enhancements to the HEPAX original added in the POWERCL.

1. *Two Multi-Function groups are implemented (FAT1 and FAT2), with a total of 154 functions distributed in two FATs as follows:*
 - a. FAT1 for the replaced YFNS, plus extra functions - with 68 entries
 - b. FAT2 for ALPHA/Buffer/XXM functionality, with 86 entries.
2. *A Catalog function is available under each of the MF groups. It can be invoked by its name (“FCAT”) or its index (00). Same name, same index (in fact same code) for the two FATs.*
 - a. CATalogs can be stopped and single-stepped (using R/S, then SST/BST).
 - b. Direct execution of the sub-function is available pressing the XEQ key in manual mode.
 - c. Section headers (function names starting with “-”) are also supported pressing “ENTER^” while in SST mode.
 - d. The listing will be printed if TRACE/NORM is set.
3. *The Numeric launchers have been merged into a single function (“XFAT __, __”) with two-parameter prompts. The first one refers to the FAT group, and the second to the index within it. So only ONE entry is used in the ROM’s main FAT.*
 - a. Using “03” or higher will be ignored during the data entry (not as after-execution error).
 - b. Additionally, using “00” in the first prompt will invoke the “XROM __, __” function.
4. *The ALPHA launchers “XEQ1” and “XEQ2” will search for sub-functions within their own FAT.*
 - a. “XEQ2” will also search within FAT1 if the function name wasn’t found in FAT2
 - b. If it’s still not found, the search will continue in the main FAT (FAT0), and even through all modules currently present on the system. “XEQ2” can be used as “universal launcher” for XROM functions and sub-functions – but not for mainframe or programs.
5. *A global XEQ launcher is also available for convenient grouping of all options.* This would be the ideal one to assign to a key, or can be called upon the main CL launcher by pressing the [SHIFT] key.
Its choices are prompted as follows: **XXEQ 0:1:2:F:R:M**
 - “0” will default to the mainframe “XEQ” function. At this point the execution is transferred to the OS, with all the same functionality available (IND, Alpha, top rows).
 - “1” will call “XEQ1”, the Alpha launcher for FAT1
 - “2” will call “XEQ2”, the Alpha launcher for FAT2, FAT1, and all main FATS present.
 - “F” will call “XFAT”, followed by the two prompts for FAT# and Fnc index.
 - “R” will call “XROM”, followed by the two prompts for XROM id# and Fnc. Id#
 - “M” will call “XQXM”, to execute a program file in XMEM which name is in ALPHA.

That's all folks, hope this is interesting readign to you all. For a real life application, just download the POWERCL to your system and take it for a spin.

To end this article, once again my unabashed thanks and acknowledgment to Steen Petersen and the original programmers of the HEPAX module - they set a new standard above and beyond the 41 OS!

Edited: 3 Sept 2012, 7:22 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Make PIL-Box ILPer program run on OS X

Posted by [Juergen Keller](#) on 2 Sept 2012, 4:24 p.m.

The PIL-Box is one of my favorite HP-41 gadgets, and my MacBook is my favorite computer, especially since it features the incredible Retina display. So it was only natural to try to make those things work together. Amazingly, I succeeded. Here is what I did.

1. First, I installed CrossOver on my MacBook. This is the commercial variant of Wine. If you don't want to spend some money, try Wine; it should work, too.
2. Then, I've [downloaded](#) and installed the OS X drivers for the FTDI chip used in the PIL-Box.
3. Next, I've opened the C: drive of my CrossOver bottle in Finder, using the bottle configuration dialog. I created a new folder in the 'Program Files' directory and copied ILPer.exe to this location.
4. As ILPer requires the MFC42.DLL, I tried to install the Microsoft Visual C++ 6 Runtime Libraries in CrossOver. Unfortunately, it didn't work. Therefore, I've downloaded the [MFC42.DLL from the Internet](#) and copied it into the bottle's C:\Windows\System32 directory, using the same procedure as for ILPer.exe.
5. Finally, I had to make the serial (USB) interface known to CrossOver. This is a little more complicated than the previous steps, as you have to execute some Terminal commands, as described in this [article in the CrossOver Support Library](#). On my system, the serial device is named /dev/cu.usbserial-FTE19YSZ, and my default CrossOver bottle is 'winxp'. So the following commands worked for me:

```
cd ~/Library/Application\ Support/CrossOver/Bottles/winxp/dosdevices  
ln -s /dev/cu.usbserial-FTE19YSZ com4
```

I'm very happy that my MacBook now speaks HP-IL, and hopefully, this description helps others to do the same.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[Return to the Index]
[Previous | Next]

HP-82163A how-to and video

Posted by [Sylvain Cote](#) on 21 July 2012, 2:47 p.m.

-
YouTube video located at : <http://youtu.be/v0yvLKpwc74>

-
HP-75C/D basic program that show the capability of the HP-82163A video interface
Used: HP-75C, I/O ROM (WKEY\$, RPT\$), HP-82163A, PIL-BOX (with Gateway & Scope soft running)

```
-
10 ! setup
20 DISPLAY IS * @ PRINTER IS ':d1' @ PWIDTH INF
30 C1=0 @ C2=1 @ C3=0 @ I1=0 @ I2=0
40 E7$=CHR$(27)
50 A$=E7$&'A' @ B$=E7$&'B' @ C$=E7$&'C' @ D$=E7$&'D'
60 E$=E7$&'E' @ H$=E7$&'H' @ J$=E7$&'J'
70 Q$=E7$&'Q' @ R$=E7$&'R' @ S$=E7$&'S' @ T$=E7$&'T'
80 L$=E7$&'<' @ G$=E7$&'>' @ P$=E7$&'%'
90 DIM S1$[96],S2$[96]
100 S1$=' ' @ FOR I=32 TO 126 @ S1$=S1$&CHR$(I) @ NEXT I
110 S2$=' ' @ FOR I=32+128 TO 126+128 @ S2$=S2$&CHR$(I) @ NEXT I
120 ! scan keyboard
125 PRINT E$&'Ready ...'
130 DISP 'Press a key ...' @ K1$=UPRC$(WKEY$) @ K2=NUM(K1$)
140 IF K2=81 THEN PRINT E$&'Leaving ...' @ END ! Q : Quit
150 IF K2=139 THEN PRINT E$; @ GOTO 130 ! CLR : device clear
160 IF K2=83 THEN PRINT S1$&S2$; @ GOTO 130 ! S : print char set
170 IF K2=65 THEN PRINT RPT$(CHR$(C3+32),32); @ C3=MOD(C3+1,94) @ GOTO 130 ! A : print a line
180 IF K2=132 THEN PRINT A$; @ GOTO 130 ! UP : cursor up
190 IF K2=133 THEN PRINT B$; @ GOTO 130 ! DOWN : cursor down
200 IF K2=134 THEN PRINT D$; @ GOTO 130 ! LEFT : cursor left
210 IF K2=135 THEN PRINT C$; @ GOTO 130 ! RIGHT : cursor right
220 IF K2=137 THEN PRINT H$; @ GOTO 130 ! FET : cursor home
230 IF K2=138 THEN PRINT J$; @ GOTO 130 ! DEL : clear screen from cursor
240 IF K2=136 AND C1=0 THEN PRINT Q$; @ C1=1 @ GOTO 130 ! I/R : show insert cursor
250 IF K2=136 AND C1=1 THEN PRINT R$; @ C1=0 @ GOTO 130 ! I/R : show replace cursor
260 IF K2=168 AND C2=0 THEN PRINT G$; @ C2=1 @ GOTO 130 ! SHIFT+I/R : cursor on
270 IF K2=168 AND C2=1 THEN PRINT L$; @ C2=0 @ GOTO 130 ! SHIFT+I/R : cursor off
280 IF K2=164 THEN PRINT S$; @ GOTO 130 ! SHIFT+UP : roll up
290 IF K2=165 THEN PRINT T$; @ GOTO 130 ! SHIFT+DOWN : roll down
300 ! G : set cursor at row,col
310 IF K2=71 THEN INPUT 'Row,Col?'; I1,I2 @ PRINT P$&CHR$(I2)&CHR$(I1); @ GOTO 130
320 GOTO 130
-----
```

YouTube video script

```
-
CLR          : clear screen
S            : print char set
A            : print a line of ' '
S            : print char set
A            : print a line of '!'
S            : print char set
A            : print a line of '""'
S            : print char set
A            : print a line of '#'
S            : print char set
A            : print a line of '$'
UP           : cursor up
```

UP : cursor up
UP : cursor up
RIGHT : cursor right
RIGHT : cursor right
RIGHT : cursor right
I/R : show insert cursor
SHIFT+I/R : cursor off
SHIFT+I/R : cursor on
I/R : show replace cursor
UP : cursor up
UP : cursor up
UP : cursor up
RIGHT : cursor right
I/R : show insert cursor
I/R : show replace cursor
SHIFT+I/R : cursor off
SHIFT+I/R : cursor on
DOWN : cursor down
LEFT : cursor left
SHIFT+UP : roll up
SHIFT+UP : roll up
SHIFT+UP : roll up
SHIFT+DOWN : roll down
SHIFT+DOWN : roll down
SHIFT+DOWN : roll down
DEL : clear screen from cursor
SHIFT+UP : roll up
SHIFT+UP : roll up
SHIFT+UP : roll up
FET : cursor home
G : set cursor at row,col : Row,Col? 10,20[RTN]
Q : quit

PIL-BOX with Gateway & Scope soft running

UNL	UNL	UNL	LAD	01	DAB	1B	DAB	45
DAB 52	DAB 65	DAB 61	DAB 64	DAB 79	DAB 20			
DAB 2E	DAB 2E	DAB 2E	DAB 0D	DAB 0A	DAB 1B			
DAB 45	DAB 20	DAB 20	DAB 21	DAB 22	DAB 23			
DAB 24	DAB 25	DAB 26	DAB 27	DAB 28	DAB 29			
DAB 2A	DAB 2B	DAB 2C	DAB 2D	DAB 2E	DAB 2F			
DAB 30	DAB 31	DAB 32	DAB 33	DAB 34	DAB 35			
DAB 36	DAB 37	DAB 38	DAB 39	DAB 3A	DAB 3B			
DAB 3C	DAB 3D	DAB 3E	DAB 3F	DAB 40	DAB 41			
DAB 42	DAB 43	DAB 44	DAB 45	DAB 46	DAB 47			
DAB 48	DAB 49	DAB 4A	DAB 4B	DAB 4C	DAB 4D			
DAB 4E	DAB 4F	DAB 50	DAB 51	DAB 52	DAB 53			
DAB 54	DAB 55	DAB 56	DAB 57	DAB 58	DAB 59			
DAB 5A	DAB 5B	DAB 5C	DAB 5D	DAB 5E	DAB 5F			
DAB 60	DAB 61	DAB 62	DAB 63	DAB 64	DAB 65			
DAB 66	DAB 67	DAB 68	DAB 69	DAB 6A	DAB 6B			
DAB 6C	DAB 6D	DAB 6E	DAB 6F	DAB 70	DAB 71			
DAB 72	DAB 73	DAB 74	DAB 75	DAB 76	DAB 77			
DAB 78	DAB 79	DAB 7A	DAB 7B	DAB 7C	DAB 7D			
DAB 7E	DAB 20	DAB A0	DAB A1	DAB A2	DAB A3			
DAB A4	DAB A5	DAB A6	DAB A7	DAB A8	DAB A9			
DAB AA	DAB AB	DAB AC	DAB AD	DAB AE	DAB AF			
DAB B0	DAB B1	DAB B2	DAB B3	DAB B4	DAB B5			
DAB B6	DAB B7	DAB B8	DAB B9	DAB BA	DAB BB			
DAB BC	DAB BD	DAB BE	DAB BF	DAB C0	DAB C1			
DAB C2	DAB C3	DAB C4	DAB C5	DAB C6	DAB C7			
DAB C8	DAB C9	DAB CA	DAB CB	DAB CC	DAB CD			
DAB CE	DAB CF	DAB D0	DAB D1	DAB D2	DAB D3			
DAB D4	DAB D5	DAB D6	DAB D7	DAB D8	DAB D9			
DAB DA	DAB DB	DAB DC	DAB DD	DAB DE	DAB DF			
DAB E0	DAB E1	DAB E2	DAB E3	DAB E4	DAB E5			
DAB E6	DAB E7	DAB E8	DAB E9	DAB EA	DAB EB			
DAB EC	DAB ED	DAB EE	DAB EF	DAB F0	DAB F1			
DAB F2	DAB F3	DAB F4	DAB F5	DAB F6	DAB F7			
DAB F8	DAB F9	DAB FA	DAB FB	DAB FC	DAB FD			
DAB FE	DAB 20	DAB 20	DAB 20	DAB 20	DAB 20			
DAB 20	DAB 20	DAB 20	DAB 20	DAB 20	DAB 20			
DAB 20	DAB 20	DAB 20	DAB 20	DAB 20	DAB 20			
DAB 20	DAB 20	DAB 20	DAB 20	DAB 20	DAB 20			
DAB 20	DAB 20	DAB 20	DAB 20	DAB 20	DAB 20			
DAB 20	DAB 20	DAB 20	DAB 20	DAB 20	DAB 21			
DAB 22	DAB 23	DAB 24	DAB 25	DAB 26	DAB 27			
DAB 28	DAB 29	DAB 2A	DAB 2B	DAB 2C	DAB 2D			
DAB 2E	DAB 2F	DAB 30	DAB 31	DAB 32	DAB 33			

DAB 34	DAB 35	DAB 36	DAB 37	DAB 38	DAB 39
DAB 3A	DAB 3B	DAB 3C	DAB 3D	DAB 3E	DAB 3F
DAB 40	DAB 41	DAB 42	DAB 43	DAB 44	DAB 45
DAB 46	DAB 47	DAB 48	DAB 49	DAB 4A	DAB 4B
DAB 4C	DAB 4D	DAB 4E	DAB 4F	DAB 50	DAB 51
DAB 52	DAB 53	DAB 54	DAB 55	DAB 56	DAB 57
DAB 58	DAB 59	DAB 5A	DAB 5B	DAB 5C	DAB 5D
DAB 5E	DAB 5F	DAB 60	DAB 61	DAB 62	DAB 63
DAB 64	DAB 65	DAB 66	DAB 67	DAB 68	DAB 69
DAB 6A	DAB 6B	DAB 6C	DAB 6D	DAB 6E	DAB 6F
DAB 70	DAB 71	DAB 72	DAB 73	DAB 74	DAB 75
DAB 76	DAB 77	DAB 78	DAB 79	DAB 7A	DAB 7B
DAB 7C	DAB 7D	DAB 7E	DAB 20	DAB A0	DAB A1
DAB A2	DAB A3	DAB A4	DAB A5	DAB A6	DAB A7
DAB A8	DAB A9	DAB AA	DAB AB	DAB AC	DAB AD
DAB AE	DAB AF	DAB B0	DAB B1	DAB B2	DAB B3
DAB B4	DAB B5	DAB B6	DAB B7	DAB B8	DAB B9
DAB BA	DAB BB	DAB BC	DAB BD	DAB BE	DAB BF
DAB C0	DAB C1	DAB C2	DAB C3	DAB C4	DAB C5
DAB C6	DAB C7	DAB C8	DAB C9	DAB CA	DAB CB
DAB CC	DAB CD	DAB CE	DAB CF	DAB D0	DAB D1
DAB D2	DAB D3	DAB D4	DAB D5	DAB D6	DAB D7
DAB D8	DAB D9	DAB DA	DAB DB	DAB DC	DAB DD
DAB DE	DAB DF	DAB E0	DAB E1	DAB E2	DAB E3
DAB E4	DAB E5	DAB E6	DAB E7	DAB E8	DAB E9
DAB EA	DAB EB	DAB EC	DAB ED	DAB EE	DAB EF
DAB F0	DAB F1	DAB F2	DAB F3	DAB F4	DAB F5
DAB F6	DAB F7	DAB F8	DAB F9	DAB FA	DAB FB
DAB FC	DAB FD	DAB FE	DAB 21	DAB 21	DAB 21
DAB 21	DAB 21	DAB 21	DAB 21	DAB 21	DAB 21
DAB 21	DAB 21	DAB 21	DAB 21	DAB 21	DAB 21
DAB 21	DAB 21	DAB 21	DAB 21	DAB 21	DAB 21
DAB 21	DAB 21	DAB 21	DAB 21	DAB 21	DAB 21
DAB 21	DAB 21	DAB 21	DAB 21	DAB 21	DAB 20
DAB 20	DAB 21	DAB 22	DAB 23	DAB 24	DAB 25
DAB 26	DAB 27	DAB 28	DAB 29	DAB 2A	DAB 2B
DAB 2C	DAB 2D	DAB 2E	DAB 2F	DAB 30	DAB 31
DAB 32	DAB 33	DAB 34	DAB 35	DAB 36	DAB 37
DAB 38	DAB 39	DAB 3A	DAB 3B	DAB 3C	DAB 3D
DAB 3E	DAB 3F	DAB 40	DAB 41	DAB 42	DAB 43
DAB 44	DAB 45	DAB 46	DAB 47	DAB 48	DAB 49
DAB 4A	DAB 4B	DAB 4C	DAB 4D	DAB 4E	DAB 4F
DAB 50	DAB 51	DAB 52	DAB 53	DAB 54	DAB 55
DAB 56	DAB 57	DAB 58	DAB 59	DAB 5A	DAB 5B
DAB 5C	DAB 5D	DAB 5E	DAB 5F	DAB 60	DAB 61
DAB 62	DAB 63	DAB 64	DAB 65	DAB 66	DAB 67
DAB 68	DAB 69	DAB 6A	DAB 6B	DAB 6C	DAB 6D
DAB 6E	DAB 6F	DAB 70	DAB 71	DAB 72	DAB 73
DAB 74	DAB 75	DAB 76	DAB 77	DAB 78	DAB 79
DAB 7A	DAB 7B	DAB 7C	DAB 7D	DAB 7E	DAB 20
DAB A0	DAB A1	DAB A2	DAB A3	DAB A4	DAB A5
DAB A6	DAB A7	DAB A8	DAB A9	DAB AA	DAB AB
DAB AC	DAB AD	DAB AE	DAB AF	DAB B0	DAB B1
DAB B2	DAB B3	DAB B4	DAB B5	DAB B6	DAB B7
DAB B8	DAB B9	DAB BA	DAB BB	DAB BC	DAB BD
DAB BE	DAB BF	DAB C0	DAB C1	DAB C2	DAB C3
DAB C4	DAB C5	DAB C6	DAB C7	DAB C8	DAB C9
DAB CA	DAB CB	DAB CC	DAB CD	DAB CE	DAB CF
DAB D0	DAB D1	DAB D2	DAB D3	DAB D4	DAB D5
DAB D6	DAB D7	DAB D8	DAB D9	DAB DA	DAB DB
DAB DC	DAB DD	DAB DE	DAB DF	DAB E0	DAB E1
DAB E2	DAB E3	DAB E4	DAB E5	DAB E6	DAB E7
DAB E8	DAB E9	DAB EA	DAB EB	DAB EC	DAB ED
DAB EE	DAB EF	DAB F0	DAB F1	DAB F2	DAB F3
DAB F4	DAB F5	DAB F6	DAB F7	DAB F8	DAB F9
DAB FA	DAB FB	DAB FC	DAB FD	DAB FE	DAB 22
DAB 22	DAB 22	DAB 22	DAB 22	DAB 22	DAB 22
DAB 22	DAB 22	DAB 22	DAB 22	DAB 22	DAB 22
DAB 22	DAB 22	DAB 22	DAB 22	DAB 22	DAB 22
DAB 22	DAB 22	DAB 22	DAB 22	DAB 22	DAB 22
DAB 22	DAB 22	DAB 22	DAB 22	DAB 22	DAB 22
DAB 22	DAB 20	DAB 20	DAB 21	DAB 22	DAB 23
DAB 24	DAB 25	DAB 26	DAB 27	DAB 28	DAB 29
DAB 2A	DAB 2B	DAB 2C	DAB 2D	DAB 2E	DAB 2F
DAB 30	DAB 31	DAB 32	DAB 33	DAB 34	DAB 35
DAB 36	DAB 37	DAB 38	DAB 39	DAB 3A	DAB 3B
DAB 3C	DAB 3D	DAB 3E	DAB 3F	DAB 40	DAB 41
DAB 42	DAB 43	DAB 44	DAB 45	DAB 46	DAB 47
DAB 48	DAB 49	DAB 4A	DAB 4B	DAB 4C	DAB 4D
DAB 4E	DAB 4F	DAB 50	DAB 51	DAB 52	DAB 53
DAB 54	DAB 55	DAB 56	DAB 57	DAB 58	DAB 59

DAB 5A	DAB 5B	DAB 5C	DAB 5D	DAB 5E	DAB 5F
DAB 60	DAB 61	DAB 62	DAB 63	DAB 64	DAB 65
DAB 66	DAB 67	DAB 68	DAB 69	DAB 6A	DAB 6B
DAB 6C	DAB 6D	DAB 6E	DAB 6F	DAB 70	DAB 71
DAB 72	DAB 73	DAB 74	DAB 75	DAB 76	DAB 77
DAB 78	DAB 79	DAB 7A	DAB 7B	DAB 7C	DAB 7D
DAB 7E	DAB 20	DAB A0	DAB A1	DAB A2	DAB A3
DAB A4	DAB A5	DAB A6	DAB A7	DAB A8	DAB A9
DAB AA	DAB AB	DAB AC	DAB AD	DAB AE	DAB AF
DAB B0	DAB B1	DAB B2	DAB B3	DAB B4	DAB B5
DAB B6	DAB B7	DAB B8	DAB B9	DAB BA	DAB BB
DAB BC	DAB BD	DAB BE	DAB BF	DAB C0	DAB C1
DAB C2	DAB C3	DAB C4	DAB C5	DAB C6	DAB C7
DAB C8	DAB C9	DAB CA	DAB CB	DAB CC	DAB CD
DAB CE	DAB CF	DAB D0	DAB D1	DAB D2	DAB D3
DAB D4	DAB D5	DAB D6	DAB D7	DAB D8	DAB D9
DAB DA	DAB DB	DAB DC	DAB DD	DAB DE	DAB DF
DAB E0	DAB E1	DAB E2	DAB E3	DAB E4	DAB E5
DAB E6	DAB E7	DAB E8	DAB E9	DAB EA	DAB EB
DAB EC	DAB ED	DAB EE	DAB EF	DAB F0	DAB F1
DAB F2	DAB F3	DAB F4	DAB F5	DAB F6	DAB F7
DAB F8	DAB F9	DAB FA	DAB FB	DAB FC	DAB FD
DAB FE	DAB 23	DAB 23	DAB 23	DAB 23	DAB 23
DAB 23	DAB 23	DAB 23	DAB 23	DAB 23	DAB 23
DAB 23	DAB 23	DAB 23	DAB 23	DAB 23	DAB 23
DAB 23	DAB 23	DAB 23	DAB 23	DAB 23	DAB 23
DAB 23	DAB 23	DAB 23	DAB 23	DAB 23	DAB 23
DAB 23	DAB 23	DAB 23	DAB 20	DAB 20	DAB 21
DAB 22	DAB 23	DAB 24	DAB 25	DAB 26	DAB 27
DAB 28	DAB 29	DAB 2A	DAB 2B	DAB 2C	DAB 2D
DAB 2E	DAB 2F	DAB 30	DAB 31	DAB 32	DAB 33
DAB 34	DAB 35	DAB 36	DAB 37	DAB 38	DAB 39
DAB 3A	DAB 3B	DAB 3C	DAB 3D	DAB 3E	DAB 3F
DAB 40	DAB 41	DAB 42	DAB 43	DAB 44	DAB 45
DAB 46	DAB 47	DAB 48	DAB 49	DAB 4A	DAB 4B
DAB 4C	DAB 4D	DAB 4E	DAB 4F	DAB 50	DAB 51
DAB 52	DAB 53	DAB 54	DAB 55	DAB 56	DAB 57
DAB 58	DAB 59	DAB 5A	DAB 5B	DAB 5C	DAB 5D
DAB 5E	DAB 5F	DAB 60	DAB 61	DAB 62	DAB 63
DAB 64	DAB 65	DAB 66	DAB 67	DAB 68	DAB 69
DAB 6A	DAB 6B	DAB 6C	DAB 6D	DAB 6E	DAB 6F
DAB 70	DAB 71	DAB 72	DAB 73	DAB 74	DAB 75
DAB 76	DAB 77	DAB 78	DAB 79	DAB 7A	DAB 7B
DAB 7C	DAB 7D	DAB 7E	DAB 20	DAB A0	DAB A1
DAB A2	DAB A3	DAB A4	DAB A5	DAB A6	DAB A7
DAB A8	DAB A9	DAB AA	DAB AB	DAB AC	DAB AD
DAB AE	DAB AF	DAB B0	DAB B1	DAB B2	DAB B3
DAB B4	DAB B5	DAB B6	DAB B7	DAB B8	DAB B9
DAB BA	DAB BB	DAB BC	DAB BD	DAB BE	DAB BF
DAB C0	DAB C1	DAB C2	DAB C3	DAB C4	DAB C5
DAB C6	DAB C7	DAB C8	DAB C9	DAB CA	DAB CB
DAB CC	DAB CD	DAB CE	DAB CF	DAB D0	DAB D1
DAB D2	DAB D3	DAB D4	DAB D5	DAB D6	DAB D7
DAB D8	DAB D9	DAB DA	DAB DB	DAB DC	DAB DD
DAB DE	DAB DF	DAB E0	DAB E1	DAB E2	DAB E3
DAB E4	DAB E5	DAB E6	DAB E7	DAB E8	DAB E9
DAB EA	DAB EB	DAB EC	DAB ED	DAB EE	DAB EF
DAB F0	DAB F1	DAB F2	DAB F3	DAB F4	DAB F5
DAB F6	DAB F7	DAB F8	DAB F9	DAB FA	DAB FB
DAB FC	DAB FD	DAB FE	DAB 24	DAB 24	DAB 24
DAB 24	DAB 24	DAB 24	DAB 24	DAB 24	DAB 24
DAB 24	DAB 24	DAB 24	DAB 24	DAB 24	DAB 24
DAB 24	DAB 24	DAB 24	DAB 24	DAB 24	DAB 24
DAB 24	DAB 24	DAB 24	DAB 24	DAB 24	DAB 24
DAB 24	DAB 24	DAB 24	DAB 24	DAB 24	DAB 1B
DAB 41	DAB 1B	DAB 41	DAB 1B	DAB 41	DAB 1B
DAB 43	DAB 1B	DAB 43	DAB 1B	DAB 43	DAB 1B
DAB 51	DAB 1B	DAB 3C	DAB 1B	DAB 3E	DAB 1B
DAB 52	DAB 1B	DAB 41	DAB 1B	DAB 41	DAB 1B
DAB 41	DAB 1B	DAB 43	DAB 1B	DAB 51	DAB 1B
DAB 52	DAB 1B	DAB 3C	DAB 1B	DAB 3E	DAB 1B
DAB 42	DAB 1B	DAB 44	DAB 1B	DAB 53	DAB 1B
DAB 53	DAB 1B	DAB 53	DAB 1B	DAB 54	DAB 1B
DAB 54	DAB 1B	DAB 54	DAB 1B	DAB 4A	DAB 1B
DAB 53	DAB 1B	DAB 53	DAB 1B	DAB 53	DAB 1B
DAB 48	DAB 1B	DAB 25	DAB 14	DAB 0A	DAB 1B
DAB 45	DAB 4C	DAB 65	DAB 61	DAB 76	DAB 69
DAB 6E	DAB 67	DAB 20	DAB 2E	DAB 2E	DAB 2E
DAB 0D	DAB 0A	LPD			

Edited: 1 July 2013, 11:27 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-41 Post Replacement

Posted by [aj04062](#) on 12 July 2012, 9:45 p.m.

The following documents a pretty straightforward method to replace broken posts on your HP-41 calculator.

Parts needed are:

- 1) Aluminum standoffs from Keystone Electronics Corp PN 1801B, Mouser PN 534-1801B
- 2) 2-56 x .25 lg Pan Head screw, Mouser PN 534-9355
- 3) 2-56 x .75 lg Pan Head screw, Mouser PN 534-9357
- 4) two part epoxy

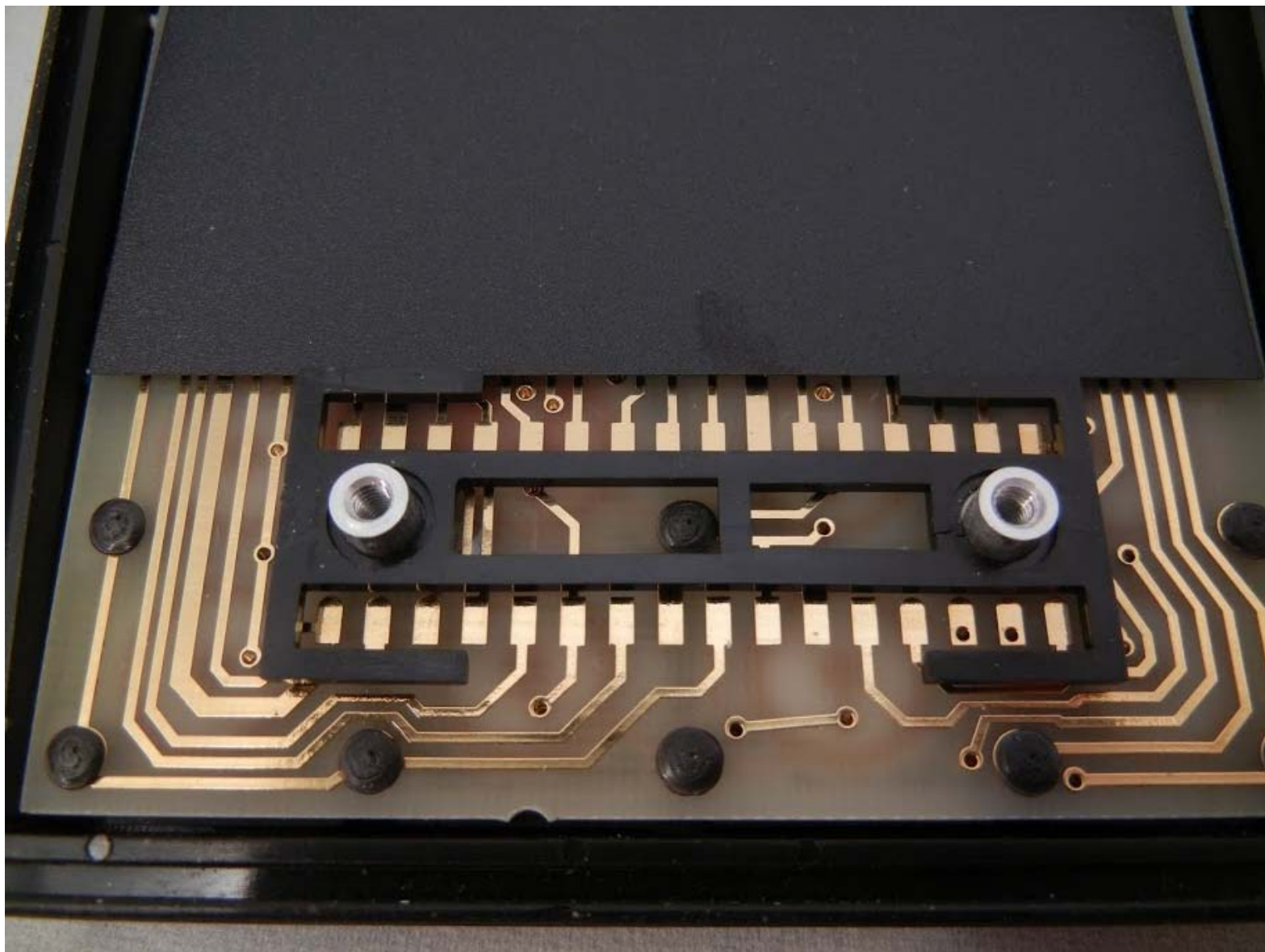
Tools needed are:

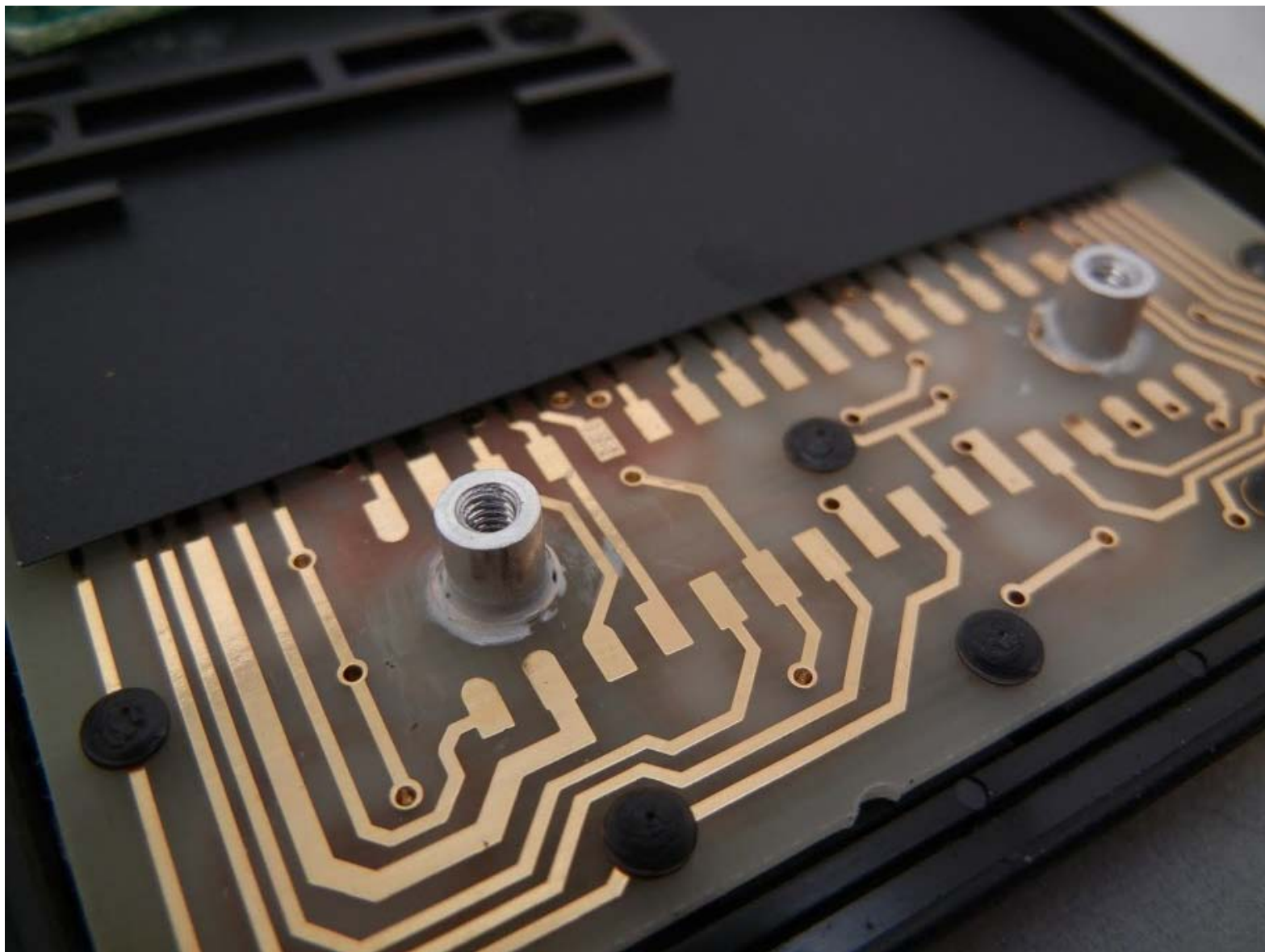
- 1) .156 diameter or equivalent drill

I started by measuring the height of the existing post. I do not have that right now, but you need to match that by drilling the hole at the right depth. I applied masking tape to the front side and to the back side to prevent chips from getting into the keyboard or places I did not want.

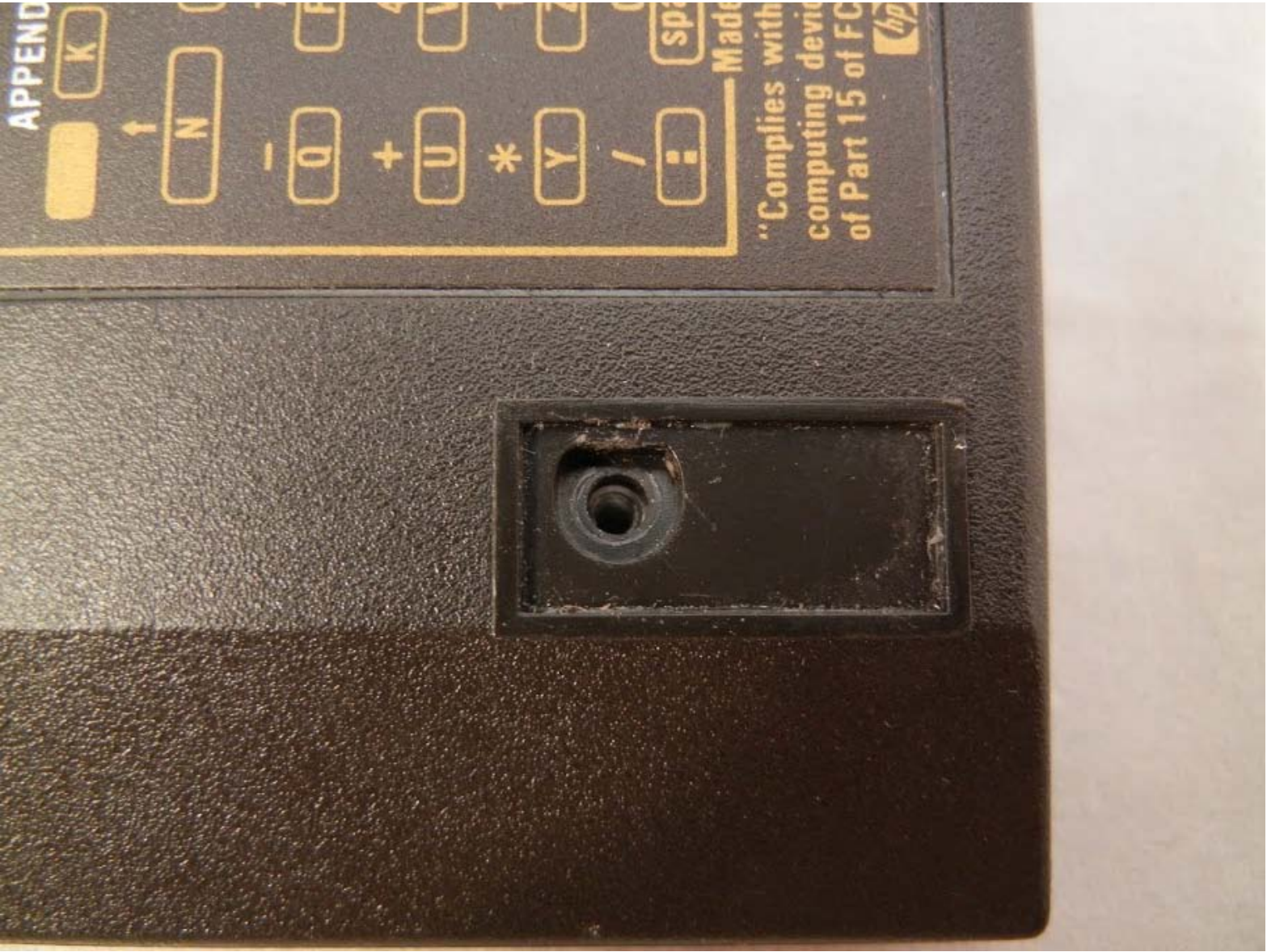
Next carefully drill out the existing posts. With a 118 degree drill tip, you can generally follow the existing hole. You shouldn't need to go much further below the board by the thickness of the PCB. If you have a drill press, this can be done a lot easier. Insert the post into the hole and check to see if the height of the post matches the old post. I have found it can be a little short. Be careful not to go too far. If you do, you can break through the keyboard overlay.

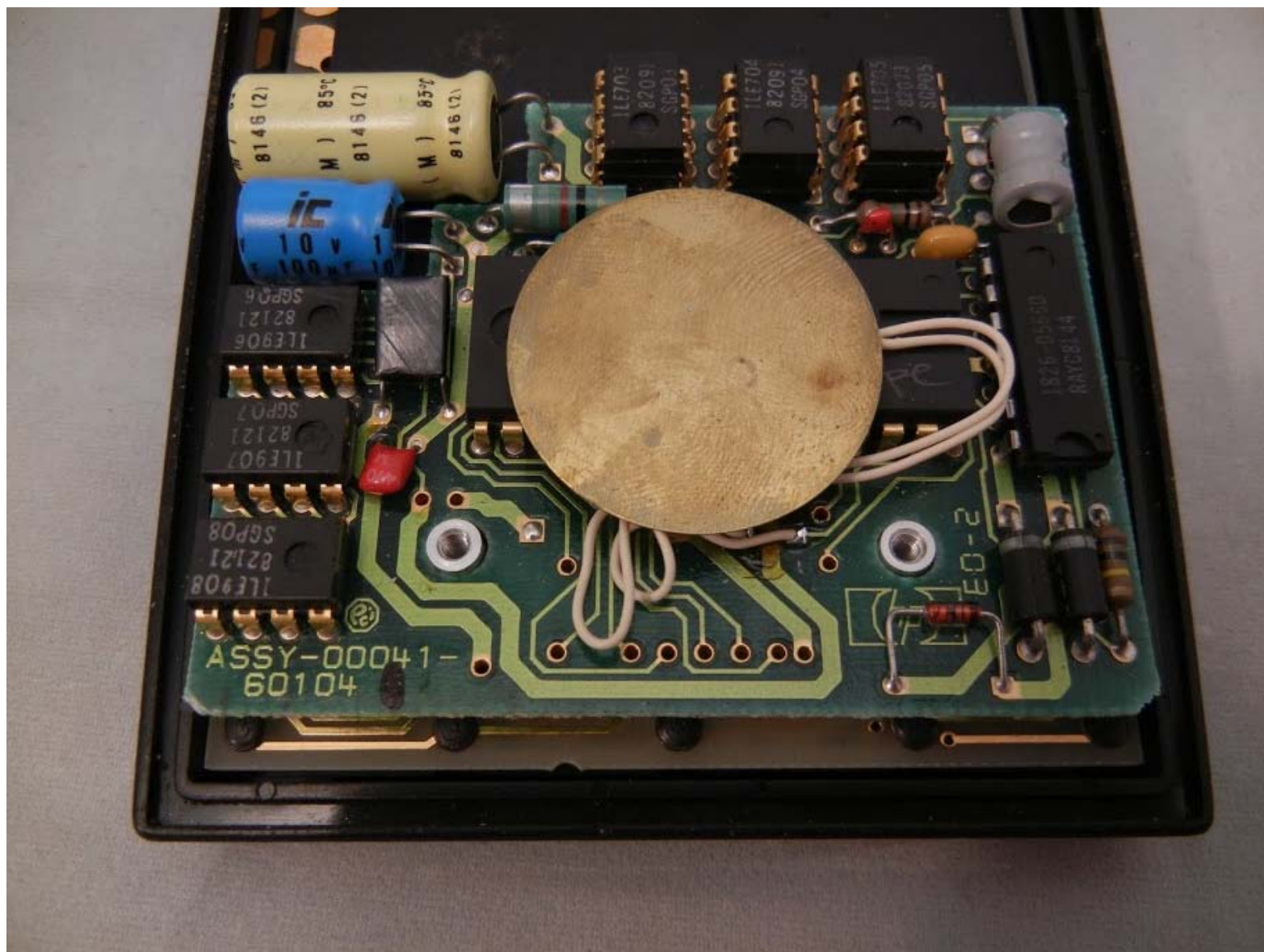
Then mix the epoxy and glue the posts in place. I installed the back and lightly screwed the #2 screws in to make sure that stayed generally aligned.











Edited: 12 July 2012, 9:48 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

41-MCODE: Dr. Jekyll & Mr. Hyde

Posted by [Ángel Martín](#) on 7 July 2012, 3:48 a.m.

I recently spent a little time experimenting with ideas to overcome the 64-function FAT limitation.

As the title suggest, this is about using alternative (or parallel) FATs for the same module (say it´s a 4k ROM). The situation is therefore that there is empty room left in the page to add more functions, but no more FAT entries are available (yes, we´ve all been here before many times...)

The idea is to use a couple of routines and their corresponding tables to re-write the FAT "on demand", thus the dual-personality concept. Such a dynamic re-writing can only be done when the module is located in RAM, be that the HEPAX, the MLDL2k, or - of course- the CL sRAM, plenty of pages there...

FAT-A would include the "MRHYDE" function, so you can switch to the FAT-B arrangement; and conversely the FAT-B would have the "DJEKYLL" routine to switch back.

This can also be implemented as an auto check upon the calculator-ON event, using the polling points to prompt which choice you want to start with. This would not require the two functions, although they´ve grown on me already, and besides they can always be used for safety.

Note that there´s no bank-switching involved. I first considered this approach but unfortunately (for this purpose but very fortunately for many others) the CL system always returns to the first bank when the calculator goes into ligh sleep, thus a hypothetical bank-2 FAT even if it existed wouldn´t hold a chance; let´s say it wouldn´t "stick" :-)

As example, let´s see how the ALPHA ROM outgrows its capabilities to have a total of 70 functions, split in 41 and 29 for each group.

The switching routines are as follows:

Header	A05B	08C	"L"	
Header	A05C	00C	"L"	
Header	A05D	019	"Y"	
Header	A05E	00B	"K"	
Header	A05F	005	"E"	
Header	A060	00A	"J"	
Header	A061	004	"D"	
DJEKIL	A062	35D	?NC XQ	Refresh C[3,6]
	A063	000	->00D7	PCTOC
	A064	09C	PT= 5	
	A065	04A	C=0 R<-	
	A066	0FA	C<>B M	Write-To adr in B[M]
	A067	0DA	C=B M	
	A068	3D0	LD@PT - F	
	A069	090	LD@PT - 2	"pF1F"
	A06A	1D0	LD@PT - 7	address of FAT-B table
	A06B	083	JNC +16d	
Header	A06C	085	"E"	
Header	A06D	004	"D"	
Header	A06E	019	"Y"	
Header	A06F	008	"H"	

Header	A070	012	"R"	
Header	A071	00D	"M"	
MRHYDE	A072	35D	?NC XQ	Refresh C[3,6]
	A073	000	->00D7	PCTOC
	A074	09C	PT= 5	
	A075	04A	C=0 R<-	clear rest of ADR field
	A076	0FA	C<>B M	Write-To adr
	A077	0DA	C=B M	keep it in C[ADR]
	A078	390	LD@PT- E	
	A079	390	LD@PT- E	"pEE8"
	A07A	210	LD@PT- 8	address of FAT-A table
	A07B	330	FETCH S&X	read source word
	A07C	0FA	C<>B M	swap addresses
	A07D	040	WROM	write word to target
	A07E	23A	C=C+1 M	increase target adr
	A07F	0FA	C<>B M	recall source adr
	A080	23A	C=C+1 M	increase source adr
	A081	2F6	?C#0 XS	end of table?
	A082	3CB	JNC -07	no, loop back
	A083	3E0	RTN	yes, done.

and the tables are just an exact copy of the respective FAT's, including the two zero words at the end, and followed by any word with a non-zero value in the XS field - which marks the end of the table to the read-write routines.

The FATS have each one extra function, to activate the parallel versions. So FAT-A includes "MRHYDE" and FAT-B includes "DJEKYLL".

The requirement imposed here is obviously the additional space required by the two tables and the switching routines (snippets really). A maximum of 128 functions (of which two are the housekeeping ones) can be accomodated, requiring 272 (2x136) words for the tables - but that's the worst-case scenario.

a FOCAL program requiring a function on the "other" FAT will simple use the "personality function" to activate it first, then back (or not depending on what comes next). To be on the safe side one can always add such call with flag 25 to trap errors in case the FAT is already the active one.

Really can't be any simpler, and it's a lot of fun to watch. Email me if you want a copy of the ALPHA44 module to play with, and by all means let's keep thinking on a way to eliminate that nasty restriction in better ways...

Cheers, AM

Edited: 9 July 2012, 9:08 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

Calculation of factorials using logarithms (HP41 MCODE)

Posted by [Frido Bohn](#) on 26 May 2012, 9:15 a.m.

Calculating Factorials in HP41 MCODE

This is a routine in MCODE I developed to overcome the situation that an HP41C can only calculate factorials of up to 69 as this is $<10^{100}$. Jean-Marc Baillard created also some great routines dealing with this "problem". The idea of this solution is to calculate the sum of logarithms from 1 to n. Thus the result of "LOGSUM" is the decimal logarithm of the factorial of n. The program takes advantage of 13-digit precision for the calculation of the natural logarithm (ln) and its sum. Eventually ln is converted to the decimal logarithm (log). The intermediate results are stored in scratch registers using MCODE routines of the operating system designed to handle 13-digit precision ([STCR] and [RCSCR]). The loop counter is also kept in a scratch register. As the result is not that vivid, a little FOCAL program is presented which converts the logarithm into an approximation of the natural number using the alpha registers. For example LOGUM 10 results in 6.5598 which is $\log(10!)$, and XFACT transforms it to 3.6288E6 ($10! = 3628800$). This program runs well on every flavor of HP41. However, as the calculations are based on an iterative process the larger the number is the longer it will last. Therefore, HP41CL in TURBO mode and emulators on fast processors have a noticeable advantage.

The execution times of XFACT at normal speed and 50x turbo and results are given below. As a reference, the truncated outputs of Wolfram Alpha are shown. (XFACT was compiled and transferred to HEPRAM. The HP-IL module was attached for printing.)

XFACT (n)	1x Speed (sec)	50x Speed (sec)	Result (truncated)
10	3.82	0.75	36288 x10^6
20	5.81	0.76	2.4329...x10^18
30	7.93	0.82	2.6525...x10^32
50	12.13	0.97	3.0414...x10^64
100	22.47	1.28	9.3326...x10^157
200	44.94	1.68	7.8865...x10^374
500	109.81	3.01	1.2201...x10^1134
1000	218.21	5.31	4.0238...x10^2567
5000	1113.30	24.25	4.2285...x10^16325
10000	n.d.	47.55	2.8462...x10^35659
100000	n.d.	473.14	2.8242...x10^456573

A linear regression of the time elapsed and the number entered shows a very good correlation coefficient $RR > 0.9999$. The ratio of the slopes of the regression line resulted to be pretty exactly 47x times larger at 50x TURBO if compared to normal speed. The small difference from 50 may be attributed to the overhead (key input, FOCAL code, display and HP-IL output). The precision deteriorates at great numbers. However, the difference is about 1% at 100000!, thus I would rather prefer the absolute difference in dollars if someone asks me.

The program accepts any non-negative integer > 0 . Theoretically, the sum of logs must be only $< 10^{100}$. Any key can

be pressed to abort the program if the user feels it lasts too long (LOGSUM would then render the iterations left to be done).

Listing of MCODE program "LOGSUM"

```

08D "M"
015 "U"
013 "S"
007 "G"
00F "O"
00C "L"      'Name of function
2A0 SETDEC    'Start (to be linked to FAT)
1A0 A=B=C=0   'Clears CPU registers A, B and C
089 ?NC XQ    'so the scratch registers Q and + get cleared to
064 ->1922    '[STSCR]
0F8 READ 3(X) 'Reads the number to be factorialized
228 WRIT 8(P) 'into register P to act as loop counter (Start of loop)
10E A=C ALL   'and into register A where MS and S&X is kept
0FA C<>B M    'while register B gets the mantissa
121 ?NC XQ    'So the 13-digit function of ln is fed correctly
06C ->1B48    '[LN13]
0D1 ?NC XQ    'Recall scratch registers Q and +
064 ->1934    '[RCSCR]
031 ?NC XQ    'add ln to the scratch using 13-digit precision
060 ->180C    '[AD2-13]
089 ?NC XQ    'and store the result back to the scratch
064 ->1922    '[STSCR]
238 READ 8(P) 'Read the loop counter into register C
10E A=C ALL   'transfer it to register A
04E C=0 ALL   'clear C
130 LDI S&X
001 CON: 1    'Load 1 into S&X of C and right shift twice to get a 1
23C RCR 2     'make CHS transformation to -1
2BE C=-C-1 MS 'As C is wrapped around the carry is set and the entry
01D ?C XQ     'point for a 10-digit addition can be accessed properly.
061 ->1807    '[AD2-10]
3CC KEY ?     'If a key is pressed the program is interrupted and shows
05F JC +&0B   'the number of loops missing for completion
2FA ?C#0 M    'Otherwise check if the counter is zero, if not go back to
34F JC -&17    'loop start
2B5 ?NC XQ    'Perform ln(10)
068 ->1AAD    '[LNC10*]
239 ?NC XQ    'Make 1/ln(10) of it
060 ->188E    '[1/X13]
0D1 ?NC XQ    'Call back the scratch content - the sum of ln
064 ->1934    '[RCSCR]
149 ?NC XQ    'and multiply it with 1/ln(10) to obtain the decimal log
060 ->1852    '[MP2-13]
0E8 WRIT 3(X) 'Put the result into stack register X
3E0 RTN       'and finish.

```

This little FOCAL program renders a legible result in decimal representation, which however cannot be used for further calculations:

Listing of FOCAL program "XFACT"

```

01*LBL "XFACT"
02 LOGSUM      'Calculate the logarithmic sum from 1 to n
03 INT         'Take the integer. It is the exponential part.
04 LASTX
05 FRC         'Take the fraction, the antilogarithm of which
06 10^X        'is the mantissa.
07 FIX 4
08 CLA
09 ARCL X
10 "|-E"       'This means APPEND "E".
11 FIX 0
12 ARCL Y
13 AVIEW       'Output as a.aaaaEbbbbbb in the alpha registers
14 FIX 4
15 END

```

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

41CL - port configuration

Posted by [Christoph Klug](#) on 22 Mar 2012, 5:14 p.m.

41CL

Logical and Physical Addresses

The logical address space consists of the ROM and RAM addresses, well known from normal 41C context. That is the 64k x 10 bit program memory = 41C ROM and the optional RAM-Box pages on pages 8 to F. Furthermore the 1k x 56 bit (7 byte) register memory = 41C RAM = Main-Memory and X-Memory. The 41C logical addresses are described in several documents, for example the CCD-Module user manual. The logical addresses of 41CL operate identically to how they work in the 41C.

In the 41CL the logical address space is mapped into a much larger physical memory space of the 41CL board which consists of internal Flash and RAM. This physical address memory space is 16M x 16 bits and is divided in half, with the lower half assigned to Flash and the upper half assigned to RAM. On Version 2 41CL boards the Flash is 1M x 16 and the RAM is 256K x 16. On Version 3 41CL boards the Flash is 2M x 16 and the RAM is 512K x 16.

In general 41CL addresses are shown in hexadecimal format which is indicated by the leading 0x. Depending on individual YFNZ commands, physical addresses are entered into the "Alpha" register as a three digit number or as a six digit number. Logical addresses are entered into the "Alpha" register as a four digit number or a one digit number - both with added bank number. Currently bank switching of RAM pages is not supported by YFNZ commands.

Using the 41CL RAM pages is similar to operating 41C RAM-Box systems from W&W, Eramco or HEPAX. Reprogramming 41CL Flash memory contents is only recommended for advanced 41CL users - please follow carefully the special hints in the 41CL manual.

Basic 41CL Software Initialisation (compatible with using HP-IL Interface Loop)

The HP41CL port configuration is done with help of the MMU = Memory Management Unit, which is programmed

by the YFNZ

- Clear MMU entries = XEQ MMUCLR
- Install YFNZ (XROM #15) to page A = "YFNZ" XEQ PLUG2L
- Enable MMU = XEQ MMUEN

page-1		/	page-2	
8		/	YFNZ	A
9		/		B
C		/	E	
D		/	F	
page-3		/	page-4	

Now YFNZ is now available on page A and page 6+7 are free for using HP-IL module

Loading CCD Module

- Install CCD Module version-B (XROM #9 + #11) to
- page 8 +9 = "CCDR" XEQ PLUG1

page-1		/	page-2	
8	CCD-1B	/	YFNZ	A
9	CCD-2B	/		B
C		/	E	
D		/	F	
page-3		/	page-4	

Loading EXT-I/O Module

- Install EXT-I/O Module version-B (XROM #23) to
- page B = "EXIO" XEQ PLUG2U

page-1		/	page-2	
8	CCD-1B	/	YFNZ	A
9	CCD-2B	/	EXT-I/O	B
C		/	E	
D		/	F	
page-3		/	page-4	

Loading EXT-IL-PLUS Module

YFNZ-2B and earlier did not include the EXT-IL+ ROM image stored in 41CL Flash, so a physical address must be used with the PLUG command. Only the three most significant physical address digits are needed. For example, if the EXT-IL+ image is loaded starting at address 0x0C8000 the following is used:

- Install EXT-IL+ Module (XROM #27) to page C = "0C8-RAM" XEQ PLUG3L

YFNZ-3A and later includes the EXT-IL+ ROM image, so the following is used:

- Install EXT-IL+ Module (XROM #27) to page C = "EILP" XEQ PLUG3L

page-1		/	page-2	
8	CCD-1B	/	YFNZ	A
9	CCD-2B	/	EXT-I/O	B
C		/		E
D	EXT-IL+			F
page-3		/	page-4	

Installing 4K RAM Pages

4K RAM pages are available in the 41CL at physical address:

0x806000 / 0x807000 / 0x808000 / 0x809000 / 0x80A000 / 0x80B000
0x80C000 / / 0x83C000 / 0x83D000 / 0x83E000 / 0x83F000

41CL hardware version 2 includes 58 available 4K RAM pages (0x806000 - 0x83F000).

41CL hardware version 3 includes 122 available 4K RAM pages (0x806000 - 0x87F000).

Before installing RAM clear the 4K block with YMCLR command as in the example below.

The YMCLR command allows the memory to be initialized with any hexadecimal value, but clearing (all zeros) the 4K block before inserting it in a Port avoids 41CL conflicts where the software might interpret the RAM contents as a module image:

"806000-0000" XEQ YMCLR

Next install the 4K page to page D = "806-RAM" XEQ PLUG3U

page-1		/	page-2	
8	CCD-1B	/	YFNZ	A
9	CCD-2B	/	EXT-I/O	B


```

-----
c  EXT-IL+ /      E
D  806-RAM /      F
-----
page-3 / page-4

```

If a 4K RAM page is installed to a port, clearing -if needed- is possible using a logical address with the command, for example = "D000-1-0000" XEQ YMCLR

To install 4K RAM to pages E and F repeat the last example with addresses 0x807000 and 0x808000.

```

page-1 / page-2
-----
8  CCD-1B / YFNZ   A
9  CCD-2B / EXT-I/O B
-----
c  EXT-IL+ / 807-RAM E
D  806-RAM / 808-RAM F
-----
page-3 / page-4

```

Loading 4K RAM Pages with HP-IL from EM41 to 41CL

Connect the 41CL with IL-Module and PIL-BOX or HP-IL/PC Interface Card to a PC for communication with EMU41. Transfer of program, data and ASCII files is possible between the 41CL and EMU41, as described in IL2000 Interface System Manual PART IX in Section III. In the same manual PART X describes handling and uploading plug in module ROM image files. ROM images, including your own individual programmed ROM solutions, can be stored on the EMU41 virtual mass storage drive. Creating and testing personal programmed ROM images inside Emu41 by using virtual RAM-BOX is more convenient than with real 41C or 41CL.

When using PIL-Box start the ILper software tool. When using HP-IL/PC Interface Card start EMU41 for HP-IL communication with the 41CL. Switch on the 41CL, set AUTOIO mode and execute SF15. With CAT 7 you list the contents of virtual mass storage drive. Enter page number 13... 15 in the X-Register and ROM image file name in "Alpha" and execute the GETROM command (which is part of the EXT-IL-Plus Module) to install the ROM images to page D, E and F.

Bank switching 4K RAM pages

It is possible to create a USER code program to perform bank switching, for any number of banks. Note that it is not necessary to use the UPLUG commands to remove an existing module or RAM from a Port. The PLUG3U / PLUG4L / PLUG4U commands automatically program the MMU to point at different RAM locations or a different ROM image.

For example, bank switching 4k RAM pages between address 0x806000 / 0x807000 / 0x808000 and address 0x809000 / 0x80A000 / 0x80B000:

01	LBL BANK1	09	LBL BANK2
02	"806-RAM"	10	"809-RAM"
03	PLUG3U	11	PLUG3U
04	"807-RAM"	12	"80A-RAM"
05	PLUG4L	13	PLUG4L
06	"808-RAM"	14	"80B-RAM"
07	PLUG4U	15	PLUG4U
08	RTN	16	END

Before using 4K RAM blocks first time, remember to clear them before installation to avoid conflicts with unknown RAM contents

Copying 4K RAM pages

This chapter describes copying the contents of a real plug in module which is placed on port 4 of 41CL to a 4K RAM page. Before installing the plug in module make pages E and F free by executing UPLUG4L and UPLUG4U (or just UPLUG4):

page-1 / page-2			
8	CCD-1B	/ YFNZ	A
9	CCD-2B	/ EXT-I/O	B

c	EXT-IL+	/	E
D	806-RAM	/	F

page-3 / page-4			

Next switch off the 41CL, insert the plug in module to port 4 and switch the 41CL on again.
In general avoid XROM number conflicts with your 41CL port configuration. Remember that 4K and 8K plug in modules exist, with one page or two pages. Handling of bank switched plug- in modules is more complex and not described here.

page-1 / page-2			
8	CCD-1B	/ YFNZ	A
9	CCD-2B	/ EXT-I/O	B

c	EXT-IL+	/ Module-lower	E
D	806-RAM	/ Module-upper	F

page-3 / page-4			

For copying the ROM images to EMU41 mass storage drive enter page number 14 or 15 in X-Register and ROM image file name in "Alpha" and execute SAVEROM command (which is part of the EXT-IL-Plus Module).

In the next example we copy the contents of the lower page of the plug in module from page E to 4k RAM on page D. We do this by operating with logical addresses entered to "Alpha" register, where -1 indicates the

bank number:

"E-1>D-1" XEQ YMCPY

Next we copy both pages of the plug in module from page E and page F to 4k RAM blocks 0x809000 and 0x80A000 by using logical- and physical addresses:

"E-1>809" XEQ YMCPY
 "F-1>80A" XEQ YMCPY

Of course YMCPY it is able to copy 4k blocks from one logical address to another logical address or from a physical address to another physical address. The YMCPY command also allows directly writing to a plug in module like a NoVRAM Module or a RAM-Box which is plugged to 41CL....

Using 41CL with I/O-Board and IL2000 Interface System

For hardware interface applications with I/O-Board or IL2000 Interface system the new 41CL would be a really powerfully and fast HP-IL controller when using the implemented TURBO modes:

I evaluate some speed tests with IL2000 Interface System and advanced audio measurement application. This includes HP-IL bidirectional data transfer between 41CL and HP82166A IL-Converter, calculations of control data and measurement values, and measurement data storage with CCD matrix files stored in X-Memory and plotting results with HP7470A IL-Plotter...

By using TURBO10 and TURBO20 I achieve a speed up in the range of 5 to 10, which is a great improvement for hardware interfacing applications with HP-41 and HP82166A IL-Converter and HP7470A IL-Plotter. The TURBO50 mode is not recommended from my side, because from this results no additional speed up improvement for this specific application.

The needed software modifications of the existing control routines, published in I/O-Board and IL 2000 Interface system manual for the 41CL are simple: With 41CL I execute data transfer between Alpha register and HP82166A IL-Converter in original HP-41 speed = TURBOX, because this transfer needs only milliseconds, compared to the program code execution which is continued during quite a view seconds. Therefore no hardware timing modification is required for I/O-Board or IL2000 Interface System.

56 LBL \$IN1	77 CLX
57 LBL 20	78 RTN
58 TURBOX	79 LBL \$OUT1
59 TRIGGER	80 LBL 22
60 X<>X	81 TURBOX
61 X<>X	82 TRIGGER
62 TRIGGER	83 2
63 2	84 OUTAN
64 INAN	85 TURBO20
65 TURBO20	86 CLX
66 CLX	87 RTN
67 RTN	88 LBL \$OUT2
68 LBL \$IN2	89 LBL 23
69 LBL 21	90 TURBOX
70 TURBOX	91 TRIGGER
71 TRIGGER	92 TRIGGER
72 TRIGGER	93 2
73 TRIGGER	94 OUTAN
74 2	95 TURBO20
75 INAN	96 CLX
76 TURBO20	97 RTN

Christoph Klug March 2012

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-41 MCODE: Checking ROMs Configuration

Posted by [Ángel Martín](#) on 18 Mar 2012, 3:23 a.m.

Problem Description.

You just plugged a module (Clonix, Standard, or "virtual" on the CL / MLDL) and your trusty 41 starts acting up when you try to execute any function. You of course suspect the amateur programmer who put it together and didn't do enough testing - but on second thought you wonder if there could be any XROM conflict in the current configuration.

Even HP-made modules suffered from this in the old days (they were marked with an "x" on the module case label) - but today is more frequent given the open access to every module on Clonix/MLDL, with the CL topping the list: a simple PLUG command for a ROM in its vast library can take you there.

Function Features.

The Function listed below determines whether such a conflict really exists. It shows a "CONFIG OK/BAD" statement to assess the status, and if there are conflicts it will sequentially show ALL conflicting XROM numbers if multiple offenders, ending with the BAD status message.

It will also discard "false positives" caused by multi-page ROMS (i.e. 2 or more 4k-blocks) with just one single FAT.

Checking Method.

Here's a crude sketch in basic to help define the context to get started. The idea is to check the first byte in pages 3 to F, looking for duplicates.

```
FOR J= 3 TO 14
W = FETCH(J)
FOR K=J+1 TO 15
V = FETCH (K)
IF V=W THEN -> BAD STUFF
NEXT K
NEXT J
```

A check for the second byte is used to determine whether the 4k page has an XROM id# but it has NO functions.

Routine Code.

Listed below is my implementation of CHKCGF. It assumes a system with the CX OS, as there are a couple of calls to the message routines.

The FOR-NEXT loop J and K indexes are stored in CPU registers N and M respectively.

The enumeration starts at page#3, thus catching the CX-XFUNCT. It is compatible with Page#4-equipped systems,

since its "fake" XROM id# is 35 - more about this on another article.

1	REPORT	215	?NC XQ	Build Msg - all cases
2		0FC	->3F85	[APRMSG2]
3		003	"C"	
4		00F	"O"	
5		00E	"N"	
6		006	"F"	
7		009	"I"	
8		007	"G"	
9		220	" "	
10		24C	?FSET 9	were there issues?
11		037	JC +06	[BADONE]
12		3BD	?NC XQ	
13		01C	->07EF	[MESSL]
14		00F	"O"	
15		20B	"K"	
16		033	JNC +06	[SHOWME]
17	BADONE	3BD	?NC XQ	
18		01C	->07EF	[MESSL]
19		002	"B"	
20		001	"A"	
21		204	"D"	
22	SHOWME	1F1	?NC GO	Show and Halt
23		0FE	->3F7C	[APEREX]
24		087	"G"	
25		006	"F"	
26		003	"C"	
27		00B	"K"	
28		008	"H"	
29		003	"C"	
30	CHKCFG	1A0	A=B=C=0	Start w/ clean slate
31		244	CLRF 9	
32		15C	PT= 6	
33		090	LD@PT- 2	puts "2" in C(6)
34		070	N=C ALL	now in N(6)
35	NEXTJ	0B0	C=N ALL	
36		15C	PT= 6	
37		222	C=C+1 @R	increases page#
38			JC -37d	[REPORT]
39		070	N=C ALL	update 1st. Counter
40		330	FETCH S&X	get 1st. Word (j)
41		106	A=C S&X	store in A[S&X]
42		23A	C=C+1 M	increase address
43		330	FETCH S&X	get 2nd. Word (j)
44		2E6	?C#0 S&X	empty FAT?
45		3B3	JNC -10d	yes, [NEXTKJ]
46		0B0	C=N ALL	bring counter back
47		222	C=C+1 @R	add one: out of bounds?
48		28F	JC -47d	yes, we're done
49		158	M=C ALL	reset 2nd. Counter
50		033	JNC +06	1st. Time is different
51	NEXTK	198	C=M ALL	
52		15C	PT= 6	
53		222	C=C+1 @R	increases page#
54		36F	JC -19d	checked all pages?
55		158	M=C ALL	update 2nd. Counter
56	1STONE	330	FETCH S&X	get 1st. Word (k)
57		0E6	C<>B S&X	store in B[S&X]
58		23A	C=C+1 M	increase address
59		330	FETCH S&X	get 2nd. Word (k)
60		2E6	?C#0 S&X	empty FAT?
61		3B3	JNC -10d	yes, -> [NEXTK]
62		0C6	C=B S&X	get 1st. Word (k)
63		366	?A#C S&X	same xrom id#'s ?
64		39F	JC -13d	no, -> [NEXTK]
65		248	SETF 9	yes, flag it as "dirty" cfg
66		215	?NC XQ	Build Msg - all cases
67		0FC	->3F85	[APRMSG2]
68		004	"D"	
69		015	"U"	
70		010	"P"	
71		020	" "	
72		018	"X"	
73		012	"R"	
74		00F	"O"	
75		00D	"M"	
76		220	" "	
77		066	A<>B S&X	value to convert in A[S&X]
78		01E	A=0 MS	

```
79      3A1      ?NC XQ      Generate number ->display!  
80      014      ->05E8      [GENNUM]  
81      031      ?NC XQ      Display not halting  
82      100      ->400C      [APERX4]  
83      37D      ?NC XQ      Cancels TURBO mode  
84      13C      ->4FDF      [TURB00]  
85      046      C=0 S&X  
86      2A6      C=-C-1 S&X Delay loop  
87      266      C=C-1 S&X  
88      3FB      JNC -01  
89      38D      ?NC XQ      Sets TURBO mode  
90      13C      ->4FE3      [TURB50]  
91      2C3      JNC -40d      [NEXTK]
```

Notes.

Lines 83-84 and 89-90 are used to remove the TURBO mode during the displaying of the messages. They're only relevant to CL systems and have no effect on other setups.

Lines 81-82 are a call to the Page#4 Library. It's for a non-halting [APEREX].

As you can see it's relatively simple and requires no deep knowledge of the 41 OS internals. I use it all the time on my CL - for obvious reasons - where I'm changing the configuration very frequently, and now that I have it I can't live without it. It's a very handy tool, I'm surprised nobody has done it before.

"AM

Edited: 19 Mar 2012, 4:31 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 17BII+ TVM Solver Solution expressing the Balloon Payment as a Percentage with Begin and End mode [and Down Payment as a percentage]

Posted by [bjmaskell](#) on 3 Mar 2012, 12:36 a.m.

HP 17BII+ Solver Solution

Description

This formula is adapted from page 199 of the Hp 17bii+ (silver) manual.

It partially recreates the TVM (Time Value of Money) solution in the solver.

It automates the balloon payment to be a percentage rather than a fixed \$dollar amount which otherwise would be calculated manually each time once the PV (loan amount) is known, by altering the FV this then affects the PMT amount. Hence the standard HP TVM function make 'what if' scenarios tedious at best when working from a fixed monthly budget scenario.

The following solver formula enables the HP17BII+ to become a 'what if' scenario calculator.

Advantages

The formula expresses the Balloon payment as a percentage rather than a \$dollar amount. Hence the [PV] loan amount and [FV] balloon payment are both calculated automatically. By altering the balloon [RESID%], and lease period [N] will estimate a maximum loan amount [PV] based upon a persons fixed monthly budget [PMT].

Example

You may wish to purchase a car on finance, you can afford \$450 per month, the finance interest is 8%, and the car dealer has a 36 month lease with up to a 40% residual, a 48 month lease up to 30% residual and a 60 month lease up to 20% residual.

Speed of Solver

You can speed up the Solver by 'guessing' - entering in a low and high guess *i.e.* \$20,000 [PV] \$50,000 [PV] [PV]

Solver Formula

Definitions:

Definitions of Key Inputs:

- PMT = Payment per period
- PV = Present Value (or loan amount)
- RSD% = Residual (or balloon payment) expressed as an Integer (i.e: 20 = 20%). RESD% is also referred to as the FV or future value expressed as a %
- DWN% = Down Payment as a percentage of the Loan Amount (PV).
- P/YR = The number of periods (or payments) per year
- I%YR = Interest Rate Per Year
- N = Length of loan expressed as a total (i.e: a 5 year loan x 12 P/YR (payments per year), N = 60)
- BEG? = When payments are made at the end of the each period enter 0 into BEG?, when payments are made at the beginning of each period enter 1 into BEG?.

Version 1 - The Original Solver Formula:

FINANCE.CALC: $PMT = (- PV - (PV \times (RSD\% \div 100)) \times (SPPV(I\%YR \div P/YR : N)) \div (USPV(I\%YR \div P/YR:N - \#ADV) + \#ADV)$

Version 2 - Improved Version

Improved Solver Formula: - This formula contributed by Peter A. Gebhardt

FINANCE.CALC: $0 \times BEG? \times N \times \#P \times I\%YR \times PV \times PMT + PMT = (-PV + PV \times (RSD\% \div 100) \times SPPV(I\%YR \div \#P:N) \div (USPV(I\%YR \div \#P:N - BEG?) + BEG?)$

Version 3 - With Downpayment as a percentage

Improved Solver Formula with ability for Down Payment as a percentage of the loan amount (PV): - This formula also contributed by Peter A. Gebhardt

RSD%, DWN% as a percentage of PV : $0 \times BEG? \times N \times \#P \times I\%YR \times PV \times DWN\% \times PMT + PMT = (-PV + PV \times DWN\% \div 100 + PV \times (RSD\% \div 100) \times SPPV(I\%YR \div \#P:N)) \div (USPV(I\%YR \div \#P:N - BEG?) + BEG?)$

Scenario Test:

Mary wishes to buy a car using a lease arrangement. What is the maximum loan amount Mary can borrow on a budget of \$450 per month. The current finance interest is 8%, and the car dealer has 3 options: (1) - a 36 month lease with up to a 40% residual; (2) - a 48 month lease up to 30% residual and; (3) - a 60 month lease up to 20% residual.

Calculate the maximum purchase price that Mary can spend on a car (Maximum Loan Amount) for each scenario.

Initial Inputs:

- P/YR = 12 (Payments per year)
- #ADV = 1 (Payments are monthly in advance)
- PMT = -450 (Buyers monthly budget amount)

Scenario 1 Inputs:

- N = 36 (Length of loan expressed as a total (i.e: a 3 year loan x 12 P/YR (payments per year))
- RESD% = 40 (Maximum residual balloon payment allowed under this loan period)
- [PV] : Result \$21,100.70 Note: You can increase the speed of the solver by entering guesses: i.e: 15000 [PV] 30000 [PV] [PV]

Scenario 2 Inputs:

- N = 48 (Length of loan expressed as a total (i.e: a 4 year loan x 12 P/YR (payments per year))
- RESD% = 30 (Maximum residual balloon payment allowed under this loan period)
- [PV] : *Result* \$23,730.89

Scenario 3 Inputs:

- N = 60 (Length of loan expressed as a total (i.e: a 5 year loan x 12 P/YR (payments per year))
- RESD% = 20 (Maximum residual balloon payment allowed under this loan period)
- [PV] : *Result* \$25,805.42

Tested on:

- HP 17BII+ (Silver)

VERSION HISTORY

UPDATE 03 MAY 2012 - IMPROVED VERSION SUGGESTION FROM Peter A. Gebhardt - THANKS PETER

There is a slight error in your formula - and a potentially misleading use of a variable name. Some cosmetics I've added as well.

Think about using BEG (or even better BEG?) instead of #ADV, because #ADV is very often used already for the number (#) of advanced payments in Leasing calculations.

The corrected formula looks like this:

$$\begin{aligned} &0 * \text{BEG?} * N * \#P * I\% \text{YR} * PV * \text{PMT} \\ &+ \text{PMT} \\ &= \\ &(-PV + PV * (\text{RSD}\% / 100) * \text{SPPV}(I\% \text{YR} / \#P : N) \\ &/ \\ &(\text{USPV}(I\% \text{YR} / \#P : N - \text{BEG?}) + \text{BEG?}) \end{aligned}$$

The "0*BEG?* ..." part gives you a chance for ordering the input values in a way conforming to the other TVM menus of the HP17bII+.

Solver Formula with Down Payment as a percentage : This formula contributed by Peter A. Gebhardt

$$\text{RSD}\%, \text{DWN}\% \text{ as a percentage of PV : } 0 * \text{BEG?} * N * \#P * I\% \text{YR} * PV * \text{DWN}\% * \text{PMT} + \text{PMT} = (-PV + PV * \text{DWN}\% \div 100 + PV * (\text{RSD}\% \div 100) * \text{SPPV}(I\% \text{YR} \div \#P : N)) \div (\text{USPV}(I\% \text{YR} / \#P : N - \text{BEG?}) + \text{BEG?})$$

Edited: 4 May 2012, 6:49 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Resurrecting a shattered HP 19B

Posted by [Neil Hamilton \(Ottawa\)](#) on 30 Jan 2012, 2:14 p.m.

The Casualty

I recently acquired a truly devastated HP 19B. When it arrived **every** heat stake in the lower section was shattered. The keyboard was lifting off the body to such an extent that the keys in the lower rows were in danger of falling out. This is in addition to the missing battery door. What a wretched life this poor machine must have had!

However, when I hooked it up to power, there was life! I had to pinch the body just below the display (and in several other places as well -- some photos show a big paper clip holding the bottom of the keyboard together -- though this was equally to keep the keys from falling out) but it was most definitely capable of life!

I am sorry I did not think to take a photo of the *before* state...

The Method

Having read Geoff Quickfall's excellent thread back in June ("[HP 28S: Shucking the Clamshell](#)"), I covered the LCD with a block of [blue Styrofoam](#) and heated the upper and lower labels with a blow drier to remove them. As with Geoff's experience, I had *a few* heat stake heads come away with the label. As I discovered later, the ones that were left in their little wells were shattered also. In fact, the only ones which were intact were the ones above the battery area.



There are 2 sizes of heat stake used in this calculator. The majority are 2.4mm (3/32") while there are 6 which are 1.6mm (1/16"). These smaller heat stakes are the 4 middle ones above the top row of keys and the 2 middle ones just below and to the right of the ENTER key (oops! I meant INPUT key). After taking out all the shattered remnants of the [heat stake heads](#), I drilled through every one using a drill press to maintain alignment. I used 0.060" and 0.0935 (ie: 3/32") bits to ensure a good, tight fit for the replacement styrene rods.

The battery compartment heat stakes were shaved conventionally (if not a bit over enthusiastically) with a brad-point bit.



Any crud left over from the drilling process had to be removed from the body cavity so the main portion of the calculator had to be fully disassembled at this point. When taking the front shell off the chassis, most of the keys fell out or *drifted* out of their assigned holes. The thin rubber membrane did **essentially nothing** to hold them in place.

It turned out that the *tight fit* for the styrene that I was hoping for was a little too tight. I had to ream each hole out by hand using the drill bits as tiny rasps. This did the trick and allowed the styrene rods to fit sufficiently well. (Note that, by mistake, I purchased 3/32" **styrene tubing instead of rods**. I don't think this made any structural difference but it would complicate matters if you wished to paint the back after you were done.)

For obvious reasons, the keyboard wants to be key-side down when the reassembly takes place. However, it turns out to be a non-trivial exercise getting the keys back in and aligning all the keyboard sandwich layers together while trying to poke the **styrene** replacement rivets into place. As others have mentioned, my state of evolution has not yet advanced to having the required 3rd (or 4th) arm necessary for such a task. None the less, I persevered and repeated the step more than a few times before I finally got everything to stick in the right place when the 2 halves were joined.



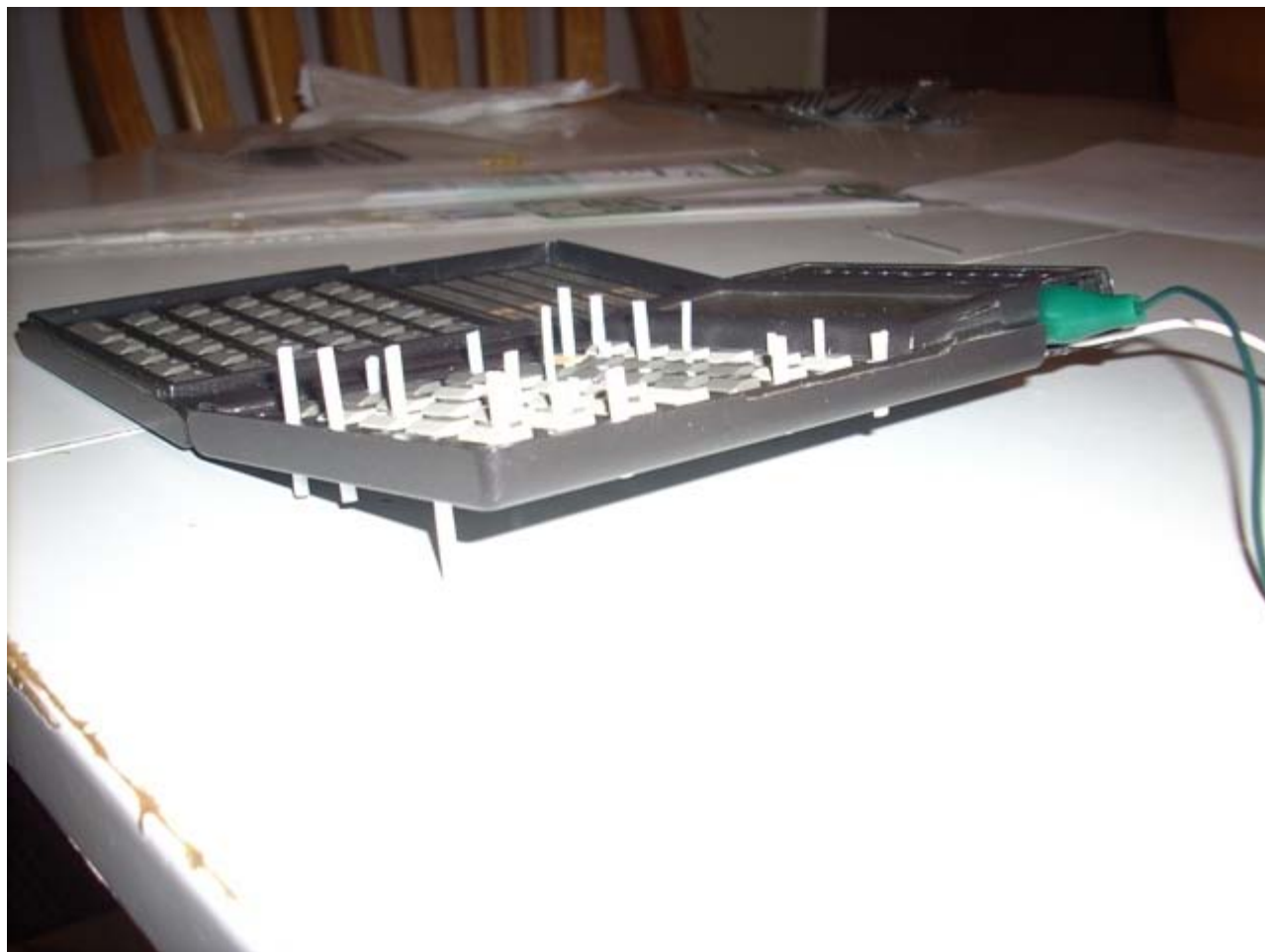


With a minimal number of the styrene rods holding the keyboard (and keys!) in place, I was able to flip the calculator over and test the connectivity and integrity of the keyboard. All keys worked acceptably except that there was very little *snap*. I have some suspicion that I may have applied too much heat to the shell in order to remove the labels. This may have compromised the snap domes somewhat (only thing I can think of).



With the keyboard integrity proven, I inserted the remainder of the styrene rods.

With the still reasonably tight fit I had created for myself, I discovered that the easiest way to get the rods through was to feed the styrene in through the back and push the whole calculator down against the table to drive the rods up through the sandwich and out the keyboard side.



I had purchased a solvent cement that worked for both the styrene and the original ABS. This was a [methylene chloride](#) based solvent and due to its low viscosity was capable of wicking down between the tightly fitted styrene rods and the holes in the back of the calculator. I bonded both the back to the styrene rods and the battery compartment heat stakes as well -- since I drilled them a little too much in the first place!

Because of the solvent's low viscosity, I did a nasty job making sure it stayed on my tiny paint brush without dripping onto the back side of the calculator. (It drips pretty easily!!) Unfortunately (or fortunately?), none of my photos does my sloppy job justice.



I next trimmed the keyboard area rods down to just above the keyboard surface, compressed the keyboard, and melted the styrene with a soldering iron.

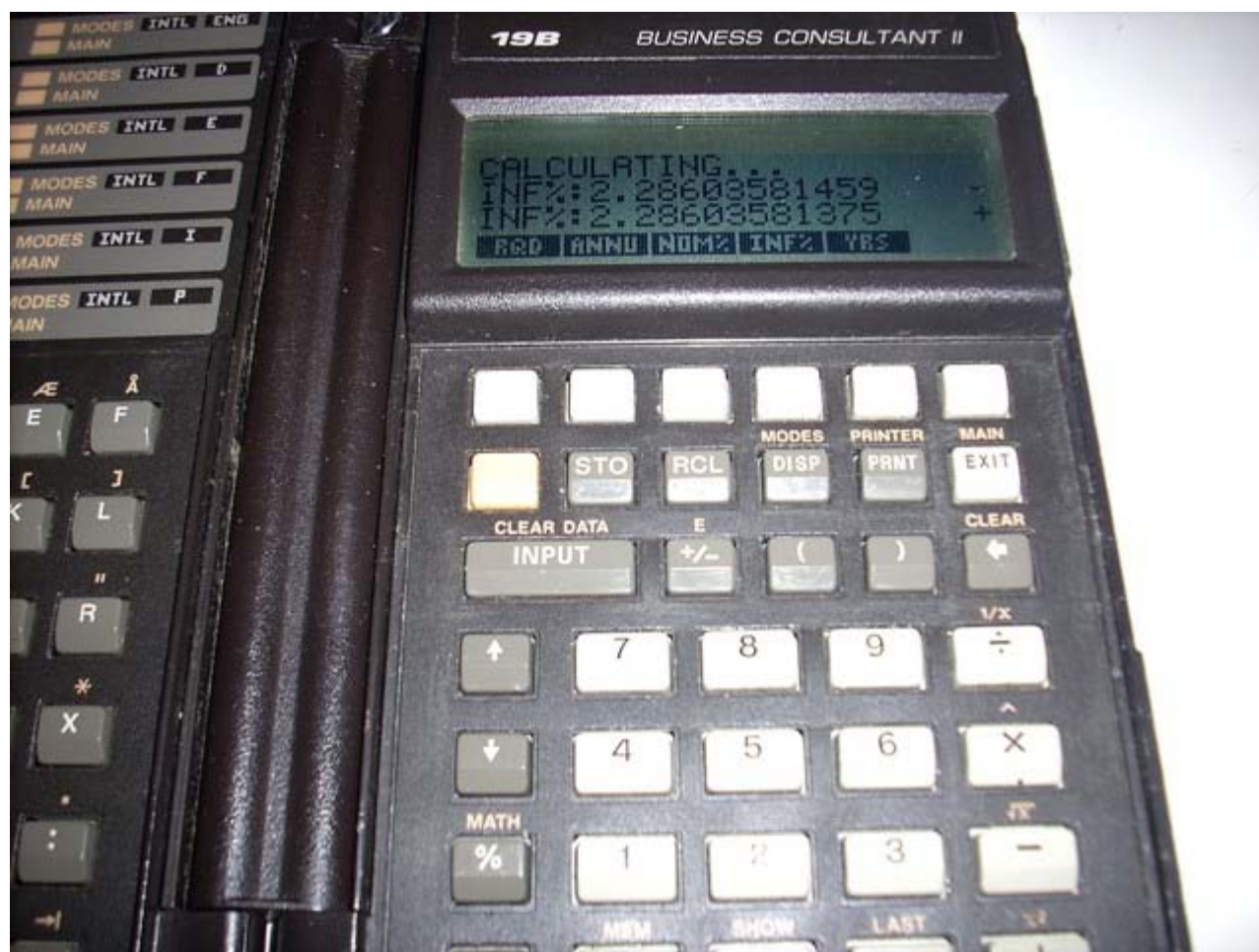
For the latter, I recalled seeing a link somewhere for a "replacement" Clamshell battery door. The following pictures show quite a reasonable, if not pretty solution to the issue.

<http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/articles.cgi?read=1150>[10/30/2014 7:40:56 PM]





It will not win any prizes for beauty but it is now functional and ready for action once again!



What I Would Do Differently

- I would use a bit less heat to remove the labels as I think this might have softened the plastic snap domes a bit.
- I would look into cutting down the battery springs somewhat before I reassembled everything so they wouldn't impart so much force to the battery door area through the batteries.
- I would buy styrene rods instead of tubes in case I wanted to paint the back afterward.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP41 Trip Computer

Posted by [Christopher Johnson](#) on 29 Jan 2012, 9:36 p.m.

Trip Computer v2.1 for HP41C/CV/CX by Christopher J. Johnson

Disclaimer:

* Requires a Time Module

* Do NOT use this program while operating your vehicle. Some states might even consider this the same as texting. Have a passanger operate the program or wait until you stop to update your odometer.

I strongly discourage this practice. User assumes all risk from the use of this program.

Introduction:

Back in the 80's I drove a car down to Florida. The vehicle was a nice Cadillac that had a trip computer. You enter your trip distance and the system would calculate miles traveled, average MPH, distance to destination and ETA. I didn't have a Cady with a trip computer so HP41 to the rescue. This program.

My goal when writing this program was small size and fast operation. I'm a machine language guy. Always looking for minimum size and fast speed.

Operation:

To start the program enter.

XEQ "TRPCOMP"

RESET??? R/S

Press R/S to start a new trip else ignore and continue to use the program with the label keys.

Starting a new trip

DIS. TO DEST.?

Enter the miles to your destination and press R/S

ODOMETER?

Enter your vehicles current odometer reading. .1 miles can be entered. Press R/S

The program will display 'COMP. STARTED' and beep at you, Then display your trip starting time

and date.

Now all that is necessary to input is your current odometer and press one of the label keys A-E. The program checks for keyboard input and recalculates if a key press was detected.

The label keys correspond to:

Distance Traveled	Elapsed Time	Average MPH	Distance to Destination	ETA
A	B	C	D	E

I hope you enjoy.

Program Listing:

```
;Registers used
; 00 = Miles Traveled
; 01 = Distance to Destination
; 02 = Starting Odometer
; 03 = Starting Time
; 04 = Current Odometer
; 05 = Current Distance To Destination
; 06 = Average MPH
; 07 = Starting Date
; 08 = Current Elapsed Time
; 09 = Current Date
; 10 = Current Time

;Labels used
; LBL A : Distance Traveled
; LBL B : Elapsed Time
; LBL C : Average Speed(MPH)
; LBL D : Distance to Destination
; LBL E : Estimated Time of Arrival(ETA)
; LBL 00 : Elapsed Time
; LBL 01 : Average Speed
; LBL 02 : Current Distance to Destination
; LBL 03 : Estimated Time of Arrival
; LBL 22 : Same Day
; LBL 33 : Reset Routine
; LBL 44 : Main Formula Routine
; LBL 55 : Past ETA Routine
; LBL 77 : Next Day ETA Routine
; LBL 99 : Common Display Routine

01 LBL "TRPCOMP"
02 FC?C 05 ;Check if this is a reset
03 GTO 33 ;Go to Reset Routine
04 SF 26 ;Set Sound Flag
05 SF 27 ;Set USER mode
06 "DIS. TO DEST.?"
07 PROMPT
08 CLRG ;A quite brutal operation, remove if offensive
09 STO 01 ;Store Distance to Destination
10 "ODOMETER?"
11 PROMPT
12 STO 02 ;Store Starting Odometer
13 DATE ;Get Current Date
14 STO 07 ;Store Starting Date
15 TIME ;Get Current Time
16 FIX 4
17 RND
18 STO 03 ;Store Starting Time
19 NOP
20 "COMP. STARTED"
21 AVIEW
22 BEEP
23 PSE
24 CLA
25 FIX 2
26 ATIME
```

```

27 X<>Y
28 ADATE
29 CF 22 ;Clear Keyboard Input Flag

30 LBL 99 ;Common Display Routine
31 CF 05 ;Clear Reset Flag
32 PROMPT

33 LBL B ;Elapsed Time
34 GTO 00

35 LBL C ;Average Speed
36 GTO 01

37 LBL D ;Current Distance to Destination
38 GTO 02

39 LBL E ;Estimated Time of Arrival
40 GTO 03

41 LBL A ;Distance Traveled
42 FS?C 22 ;Check for Keyboard Input
43 XEQ 44 ;Go To Main Formula Routine
44 RCL 00 ;Miles Traveled
45 FIX 1
46 "MTV: "
47 ARCL X
48 GTO 99 ;Go To Common Display Routine

49 LBL 00 ;Elapsed Time Display
50 FS?C 22 ;Check for Keyboard Input
51 XEQ 44 ;Go To Main Formula Routine
52 RCL 08 ;Current elapsed Time
53 FIX 4
54 "ELAP:"
55 ATIME24
56 GTO 99 ;Go To Common Display Routine

57 LBL 01 ;Average Speed
58 FS?C 22 ;Check for Keyboard Input
59 XEQ 44 ;Go To Main Formula Routine
60 RCL 06 ;Recall Average MPH
61 FIX 1
62 "AVG. MPH="
63 ARCL X
64 GTO 99 ;Go To Common Display Routine

65 LBL 02 ;Distance to Destination
66 FS?C 22 ;Check for Keyboard Input
67 XEQ 44 ;Go To Main Formula Routine
68 RCL 05 ;Recall Miles to Destination
69 FIX 1
70 "DEST: "
71 X=0? ;Check if We Have Reached Destination
72 BEEP
73 X<0? ;Check if We Have Passed Destination
74 TONE 9
75 ARCL X
76 GTO 99 ;Go To Common Display Routine

77 LBL 03 ;Estimated Time of Arrival
78 FS?C 22 ;Check for Keyboard Input
79 XEQ 44 ;Go To Main Formula Routine
80 RCL 05 ;Recall Miles to Destination
81 X<0? ;Check if we Have Passed the Destination
82 GTO 55
83 RCL 06 ;Recall Average MPH
84 / ;Calculate Travel Time to Destination
85 HMS
86 RCL 10 ;Recall Current Time

```



```

87 HMS+
88 24
89 X<Y? ;Check if ETA will be the Next Day
90 GTO 77 ;Go To ETA Next Day
91 FIX 2
92 X<>Y
93 "ETA:"
94 ATIME
95 GTO 99 ;Go To Common Display Routine

96 LBL 44 ;Main Formula Routine
97 STO 04 ;Store Current Odometer
98 RCL 02 ;Recall Starting Odometer
99 -
100 STO 00 ;Store Miles Traveled
101 RCL 01 ;Recall Distance to Destination
102 X<>Y
103 -
104 STO 05 ;Store Current Distance to Destination
105 TIME
106 STO 10 ;Store Current Time
107 DATE
108 STO 09 ;Store Current Date
109 RCL 07 ;Recall Starting Date
110 X<>Y
111 X=Y?
112 GTO 22 ;Go To Same Day
113 DDATE
114 24
115 *
116 RCL 10 ;Recall Current Time
117 HMS+
118 RCL 03 ;Recall Starting Time
119 HMS-
120 STO 08 ;Store Current Elapsed Time
121 RCL 00 ;Recall Miles Traveled
122 X<>Y
123 HR
124 /
125 STO 06 ;Store Average MPH
126 RTN

127 LBL 22 ;Same Day
128 RCL 10 ;Recall Current Time
129 RCL 03 ;Recall Starting Time
130 HMS-
131 STO 08 ;Store Current Elapsed Time
132 RCL 00 ;Recall Miles Traveled
133 X<>Y
134 HR
135 /
136 STO 06 ;Store Average MPH
137 RTN

138 LBL 77 ;ETA Next Day
139 - ;Subtract 24 Hours From Time
140 RCL 09 ;Recall Current Date
141 1 ;Add 1 Day to Starting Date
142 DATE+
143 CLA
144 FIX 2
145 ADATE
146 "T:"
147 X<>Y ;Recall Time From Y
148 ATIME
149 GTO 99

150 LBL 55 ;Past ETA Routine
151 "ETA: PAST ETA"
152 TONE 0
153 GTO 99

154 LBL 33 ;Reset Routine
155 SF 05
156 "RESET??? R/S"

```

```
157 AVIEW
158 "2.1"
159 END
```

:Updated listing

Edited: 21 Feb 2012, 5:03 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

[WP34s] The ultimate Polynomial RootSolver for the WP34s

Posted by [fhub](#) on 3 Jan 2012, 7:23 a.m.

Program removed!

Edited: 15 June 2012, 5:11 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

Go back to the main exhibit hall



HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

RPN Sunrise / Sunset programs for HP41C/HP42S/HP33S/HP35S

Posted by [M. Joury](#) on 14 Dec 2011, 9:25 a.m.

Following are a set of sunrise/sunset programs that I have developed for the HP41/HP42/HP33S/HP35S calculators. Comments are suggestions are welcome.

HP-41C and HP42S (with modifications to date routines) program to display sunrise / sunset for a given date

Usage:

Using local labels:

A = sunrise sunset for today (not available on HP42S since this routine assumes that availability of a Time module)
 B = sunrise / sunset for tomorrow (not available on HP42S since this routine assumes that availability of a Time module)
 C = sunrise / sunset for date in X register. Date should be provided in current DMY / MDY format either as dd.mmyyyy or as mm.ddyyyy respectively. Correct handling for dates in either format is dependent on the DOY routine presented in this article.
 D = Flip daylight savings time mode. Flag 01 clear = standard time / Flag 01 set = daylight savings time.
 E = Flip display mode. Flag 00 clear = single line display / Flag 00 set = multiline display

After running the program the following results are on the stack.

Z: Sunset time
 Y: Sunrise time
 X: Length of day

=====

Main HP41C program

```

01      LBL  SUN
02      LBL  A   Local label execute for current date
03      DATE    Get current date
04      GTO  03  Jump to main routine
05      LBL  B   Local label execute for current date + 1 day
06      DATE    Get current date
07      1
08      DATE+   Add 1 day to current date
09      LBL  C   Local label execute for date in X register
10      LBL  03  Used for main routine for performance reasons
11      DEG     Set degree mode
12      FIX  2   Set FIX with 2 decimal places (HH:MM)
13      CF  02   Clear sunrise / sunset setting (calc sunrise first)
14      XEQ  'DOY   Get day of year. Separate routine
15      STO  05   Store day of year
    
```

```

16      -5      UTC offset
17      ST0 03
18      90.83333333 Zenith value for sunrise. Change for twilight times
19      ST0 06
20      42.24157 Latitude. Change to match location
21      ST0 01
22      -83.75537 Longitude (west is negative). Change to match location
23      ST0 02
24      15
25      / Longitude Hour Angle
26      ST0 07
27      XEQ 01 Compute sunrise / sunset depending on flag 02
28      ST0 15 Save first computation (sunrise / sunset)
29      SF 02 Set to compute sunset
30      XEQ 01 Compute sunrise / sunset depending on flag 02
31      RCL 15 Place first result in stack (Y: sunset / X: sunrise)
32      FS 00 Which display routine has been requested?
33      GTO 00 Go to multiline routine
34      CLA Clear ALPHA register
35      ATIME Take time from X and format into ALPHA (sunrise)
36      }-R:S- Append '-R:S-' to ALPHA
37      X<>Y Get second result (Y: sunrise / X: sunset)
38      ATIME Take time from X and format into ALPHA (sunset)
39      X<>Y Restore initial order (Y: sunset / X: sunrise)
40      RCL Y Recall sunset
41      RCL Y Recall sunrise
42      HMS- Compute day length (Z: sunset / Y: sunrise / X: length)
43      BEEP Alert that we have an answer
44      AVIEW Display result as a single line (no day length)
45      RTN Backspace to see day length in the X register
46      LBL 00 Multiline display routine
47      SF 21 Set to pause on AVIEW when no printer present
48      'RISE: '
49      ATIME Append sunrise time
50      AVIEW
51      'SET: '
52      X<>Y Bring sunset into X
53      ATIME Append sunset time
54      X<>Y Restore original order
55      AVIEW
56      RCL Y Recall sunset
57      RCL Y Recall sunrise
58      HMS-
59      'LENGTH: '
60      ATIME Append day length
61      AVIEW
62      RTN
63      LBL 01 Main sunrise / sunset routine
64      6
65      FS? 02 Clear = compute sunrise / set = compute sunset
66      18
67      RCL 07
68      .
69      24
70      /
71      RCL 05
72      +
73      ST0 08
74      .9856
75      x
76      3.289
77      .
78      ST0 09
79      SIN
80      1.916
81      x
82      RCL 09
83      +
84      2
85      RCL 09
86      x
87      SIN
88      .02
89      x
90      +
91      282.634
92      +
93      360
94      MOD
95      ST0 10
96      TAN

```



```

97      .91764
98      x
99      ATAN
100     360
101     MOD
102     STO 11
103     RCL 10
104     90
105     /
106     XEQ 'FLOOR      Floor function.  External program
107     90
108     x
109     +
110     RCL 11
111     90
112     /
113     XEQ 'FLOOR      Floor function.  External program
114     90
115     x
116     •
117     15
118     /
119     STO 11
120     RCL 10
121     SIN
122     .39782
123     x
124     STO 12
125     ASIN
126     COS
127     STO 13
128     RCL 06
129     COS
130     RCL 01
131     SIN
132     RCL 12
133     x
134     •
135     RCL 01
136     COS
137     RCL 13
138     x
139     /
140     ACOS
141     FS?C 02          Clear = compute sunrise / set = compute sunset
142     GT0 02
143     360
144     X<>Y
145     •
146     LBL 02
147     15
148     /
149     RCL 11
150     +
151     .06571
152     RCL 08
153     x
154     •
155     6.622
156     •
157     24
158     MOD
159     RCL 07
160     •
161     RCL 03
162     +
163     FS? 01 Check for daylight savings
164     1
165     FS? 01 Check for daylight savings
166     +
167     HMS
168     RTN
169     LBL D Set daylight savings mode.  Flag 01 set when in effect
170     FC?C 01
171     SF 01
172     RTN
173     LBL E Set display mode.
174     FC?C 00          F00 Clear = single line / F00 set = multiline
175     SF 00
176     END

```

=====

CEIL and FLOOR functions used by the HP41C version. You only need the FLOOR code

```

01      LBL  'CEIL
02      INT
03      X<>Y
04      X<>  L
05      X>Y?
06      ISG  Y
07      LBL  00
08      X<>  L
09      X<>Y
10      RTN
11      LBL  'FLOOR
12      INT
13      X<>Y
14      X<>  L
15      X<Y?
16      DSE  Y
17      LBL  00
18      X<>  L
19      X<>Y
20      END

```

=====

DOY (Day Of Year) routine used by the HP41C version. Note that as mentioned elsewhere the 41C version assumes that a Time module is available.

```

01      LBL  'DOY
02      ENTER^
03      ENTER^
04      1  E2
05      X
06      FRC
07      .0001
08      .
09      FC?  31 MDY mode
10      1231
11      FS?  01 DMY mode
12      3112
13      +
14      1  E2
15      /
16      X<>Y
17      DDAYS
18      END

```

=====

Date calculation routines for HP42S. Can also be used for an HP41C or CV without a Time module.

A call to DOY with the date (MM.DDYYYY) in the X register returns the day of the year.
A call to DDAYS with date in Y and X registers returns the number of days between dates.

```

01      LBL  "DtCalc"
02      LBL  "DOY"
03      LBL  A

```

The following two lines can be replaced with
RCL ST X

```

04      ENTER  Push date onto stack
05      ENTER
06      1E2     Get rid of MM.DD
07      x      by shifting left and keeping
08      FP     the floating portion (year)
09      0.0001 Year = year - 1
10      -
11      1E2     Shift year back to the right

```

```

12      /
13      12.31      Add 12.31 to make this the
14      +          last day of the previous year
15      X<>Y
16      LBL      "DDAYS"
17      LBL      B
18      ST0      "DATE"
19      X<>Y
20      XEQ      90 Get day number for first date
21      ST0      01
22      RCL      "DATE"
23      XEQ      90
24      RCL -    01
25      RTN
26      LBL      "DN"
27      LBL      D
28      LBL      90
29      ENTER
30      IP
31      9
32      +
33      12
34      MOD
35      ST0      "MONTH"
36      X<>Y
37      ENTER
38      FP
39      1E2
40      X
41      IP
42      ST0      "DAY"
43      RDN
44      1E2
45      X
46      FP
47      1E4
48      X
49      RCL      "MONTH"
50      10
51      /
52      IP
53      -
54      ST0      "YEAR"
55      365
56      X
57      RCL      "YEAR"
58      4
59      /
60      IP
61      +
62      RCL      "YEAR"
63      1E2
64      /
65      IP
66      -
67      RCL      "YEAR"
68      400
69      /
70      IP
71      +
72      RCL      "MONTH"
73      306
74      X
75      5
76      +
77      10
78      /
79      IP
80      +
81      RCL      "DAY"
82      1
83      -
84      +
85      END

```

=====

Flag and register usage and notes.

Flag	Purpose	Comments
00	Choose display mode	Clear = single line / Set = multi line
01	Daylight savings time	Clear = standard time / Set = DST
02	Sunrise or sunset	Clear = compute sunrise / Set = sunset

Register	Usage
33S / 35S / 39gs	41C / 42S
A	01 Latitude
O	02 Longitude
U	03 Offset from UTC (local time)
D	04 Date (not really required)
N	05 Days from the beginning of the year.
Z	06 Zenith
G	07 Longitude -> Hour Angle (lngHour)
T	08 Approximate time (t)
M	09 Sun's mean anomaly (M)
L	10 Sun's true longitude (L)
R	11 Right Ascension (RA)
S	12 Sun's declination (sinDec)
C	13 Sun's declination (cosDec)
H	14 Sun's local hour angle (H)
B	15 Stores first result (rise or set)

Zenith for Sunrise Values		
Official Sunrise	90 degrees 50'	90.833333333333
Civil Twilight	96 degrees	96.0000
Nautical Twilight	102 degrees	102.0000
Astronomical Twilight	108 degrees	108.0000

Some Lat / Long Values		
City	Latitude	Longitude
Albuquerque, NM	35.18066	-106.49527
Ann Arbor, MI	42.24157	-83.75537

Notes:

The HP-41C version depends on having the Time Module available for full functionality.

This version has a few tweaks that are not available in the 33S / 35S version. These are as follows:

1. Sunrise / sunset for today (LBL A)
2. Sunrise / sunset for tomorrow (LBL B)
3. Two different display options
4. Day length
5. Daylight savings time without a code (UTC offset) change

If desired item 4 can be easily retrofitted to the 33S and 35S versions. However, it is probably easier to simply have the user manually set a flag rather than have a short routine to do so.

If desired this program can be cut back to the functionality of the 33S and 35S versions. This might be desirable if memory space is at a premium. The features above can all be removed relatively easily.

1. Sunrise / sunset for today: Remove LBL A. Code between lines 02 and 04 inclusive.
 2. Sunrise / sunset for tomorrow: Remove LBL B. Code between lines between 05 and 08 inclusive.
 3. If both changes above are made LBL 03 (line 10) can be removed. With these changes it might also make sense to replace LBL C with LBL A or remove local labels completely.
 4. The logic controlling the display of results is contained in lines 32-61 inclusive. Choose either approach or neither and just leave the results on the stack. The logic that calculates the day length is also included in this code. This is calculated in both display routines.
- If this logic is to be retained keep either the set of lines 40-42 or the lines 56-58.

5. If one or both display options are removed also remove the logic that switches between them.
This is LBL E and can be found at lines 173-175. It is also possible to keep the display options but remove this code and set flag 00 manually.
6. If the logic that handles Daylight Savings Time is not needed it can also be removed.
The code can be found at lines 163-166.
7. If the DST logic is removed also remove the logic that switches between standard time and DST.
This is LBL D and can be found at lines 169-172. As with the display option selection code this selection code can be removed and flag 01 can be set manually.

A further optional improvement may be added. If you want to see the date for which sunrise / sunset times are being calculated add the following code right after LBL 03:

```
11      CF 21  Don't stop on AVIEW when printer is absent
12      CLA      Clear ALPHA register
13      ADATE    Format and append date from X into ALPHA
14      AVIEW    View date during execution
```

=====

Following are versions for the HP33S and HP35S. Note that both version require the date routines at the bottom of this article.

=====

Programs for HP33S single label followed by short form

Single label

```
C0001  LBL  C
C0002  DEG
C0003  CF  0
C0004  XEQ  X
C0005  STO  N
C0006  -5
C0007  STO  U
C0008  90.83333333
C0009  STO  Z
C0010  42.24157
C0011  STO  A
C0012  -83.75537
C0013  STO  0
C0014  15
C0015  /
C0016  STO  G
C0017  SF  0  If flag 0 is set we are calculating sunset
C0018  XEQ  C028  Calculate sunset first
C0019  STO  B  Save sunset time
C0020  CF  0
C0021  XEQ  C028
C0022  RCL  B
C0023  SF  10
C0024  SUNRISE-SUNSET
C0025  CF  10
C0026  CF  0
C0027  RTN
C0028  6
C0029  FS?  0  If set use 18 for sunset calculation
C0030  18
C0031  RCL-  G
C0032  24
C0033  /
C0034  RCL+  N
C0035  STO  T
C0036  0.9856
C0037  x
C0038  3.289
C0039  1.
C0040  STO  M
C0041  SIN
C0042  1.916
C0043  x
```

```

C0044 RCL+ M
C0045 2
C0046 RCLx M
C0047 SIN
C0048 0.02
C0049 x
C0050 +
C0051 3
C0052 RCLx M
C0053 SIN
C0054 0.0003
C0055 x
C0056 +
C0057 282.634
C0058 +
C0059 360
C0060 RMDR
C0061 STO L
C0062 TAN
C0063 0.91764
C0064 x
C0065 ATAN
C0066 360
C0067 RMDR
C0068 STO R
C0069 RCL L
C0070 90
C0071 /
C0072 INTG
C0073 90
C0074 x
C0075 +
C0076 RCL R
C0077 90
C0078 /
C0079 INTG
C0080 90
C0081 x
C0082 1.
C0083 15
C0084 /
C0085 STO R
C0086 RCL L
C0087 SIN
C0088 0.39782
C0089 x
C0090 STO S
C0091 ASIN
C0092 COS
C0093 STO C
C0094 RCL Z
C0095 COS
C0096 RCL A
C0097 SIN
C0098 RCLx S
C0099 1.
C0100 RCL A
C0101 COS
C0102 RCLx C
C0103 /
C0104 ACOS
C0105 FS? 0
C0106 GT0 C110
C0107 360
C0108 X<>Y
C0109 1.
C0110 15
C0111 /
C0112 RCL+ R
C0113 0.06571
C0114 RCLx T
C0115 1.
C0116 6.622
C0117 1.
C0118 24
C0119 RMDR
C0120 RCL - G
C0121 RCL+ U
C0122 ->HMS
C0123 RTN

```

These lines can be removed from both
this program and the single label version
they have an extremely small influence on
the result.

Short form

```

C0001  LBL  C
C0002  DEG
C0003  CF  0
C0004  XEQ  X
C0005  STO  N
C0006  -5
C0007  STO  U
C0008  90.83333333
C0009  STO  Z
C0010  42.24157
C0011  STO  A
C0012  -83.75537
C0013  STO  0
C0014  15
C0015  /
C0016  STO  G
C0017  SF  0   If flag 0 is set we are calculating sunset
C0018  XEQ  V   Calculate sunset first
C0019  STO  B   Save sunset time
C0020  CF  0
C0021  XEQ  V
C0022  RCL  B
C0023  SF  10
C0024  SUNRISE-SUNSET
C0025  CF  10
C0026  CF  0
C0027  RTN
V0001  LBL  V
V0002  6
V0003  FS?  0   If set use 18 for sunset calculation
V0004  18
V0005  RCL-  G
V0006  24
V0007  /
V0008  RCL+  N
V0009  STO  T
V0010  0.9856
V0011  x
V0012  3.289
V0013  1.
V0014  STO  M
V0015  SIN
V0016  1.916
V0017  x
V0018  RCL+  M
V0019  2
V0020  RCLx  M
V0021  SIN
V0022  0.02
V0023  x
V0024  +
V0025  3
V0026  RCLx  M   These lines can be removed from both
V0027  SIN      this program and the single label version
V0028  0.0003   they have an extremely small influence on
V0029  x         the result.
V0030  +
V0031  282.634
V0032  +
V0033  360
V0034  RMDR
V0035  STO  L
V0036  TAN
V0037  0.91764
V0038  x
V0039  ATAN
V0040  360
V0041  RMDR
V0042  STO  R
V0043  RCL  L
V0044  90
V0045  /
V0046  INTG
V0047  90
V0048  x

```

```

V0049 +
V0050 RCL R
V0051 90
V0052 /
V0053 INTG
V0054 90
V0055 x
V0056 1.
V0057 15
V0058 /
V0059 STO R
V0060 RCL L
V0061 SIN
V0062 0.39782
V0063 x
V0064 STO S
V0065 ASIN
V0066 COS
V0067 STO C
V0068 RCL Z
V0069 COS
V0070 RCL A
V0071 SIN
V0072 RCLx S
V0073 1.
V0074 RCL A
V0075 COS
V0076 RCLx C
V0077 /
V0078 ACOS
V0079 FS? 0
V0080 GT0 W
V0081 360
V0082 X<>Y
V0083 1.
W0001 LBL W
W0002 15
W0003 /
W0004 RCL+ R
W0005 0.06571
W0006 RCLx T
W0007 1.
W0008 6.622
W0009 1.
W0010 24
W0011 RMDR
W0012 RCL - G
W0013 RCL+ U
W0014 ->HMS
W0015 RTN

```

=====

Programs for HP35S short form. Single label version is identical to HP33S single label

```

C0001 LBL C
C0002 DEG
C0003 CF 0
C0004 XEQ X
C0005 STO N
C0006 -5
C0007 STO U
C0008 90.83333333
C0009 STO Z
C0010 42.24157
C0011 STO A
C0012 -83.75537
C0013 STO 0
C0014 15
C0015 /
C0016 STO G
C0017 SF 0 If flag 0 is set we are calculating sunset
C0018 XEQ C028 Calculate sunset first
C0019 STO B Save sunset time
C0020 CF 0
C0021 XEQ C028
C0022 RCL B
C0023 SF 10

```

```

C0024  SUNRISE-SUNSET
C0025  CF  10
C0026  CF  0
C0027  RTN
C0028  6
C0029  FS?  0  If set use 18 for sunset calculation
C0030  18
C0031  RCL-  G
C0032  24
C0033  /
C0034  RCL+  N
C0035  STO  T
C0036  0.9856
C0037  x
C0038  3.289
C0039  1.
C0040  STO  M
C0041  SIN
C0042  1.916
C0043  x
C0044  RCL+  M
C0045  2
C0046  RCLx  M
C0047  SIN
C0048  0.02
C0049  x
C0050  +
C0051  3
C0052  RCLx  M  These lines can be removed from both
C0053  SIN      this program and the single label version
C0054  0.0003  they have an extremely small influence on
C0055  x        the result.
C0056  +
C0057  282.634
C0058  +
C0059  360
C0060  RMDR
C0061  STO  L
C0062  TAN
C0063  0.91764
C0064  x
C0065  ATAN
C0066  360
C0067  RMDR
C0068  STO  R
C0069  RCL  L
C0070  90
C0071  /
C0072  INTG
C0073  90
C0074  x
C0075  +
C0076  RCL  R
C0077  90
C0078  /
C0079  INTG
C0080  90
C0081  x
C0082  1.
C0083  15
C0084  /
C0085  STO  R
C0086  RCL  L
C0087  SIN
C0088  0.39782
C0089  x
C0090  STO  S
C0091  ASIN
C0092  COS
C0093  STO  C
C0094  RCL  Z
C0095  COS
C0096  RCL  A
C0097  SIN
C0098  RCLx  S
C0099  1.
C0100  RCL  A
C0101  COS
C0102  RCLx  C
C0103  /
C0104  ACOS

```

```

C0105    FS?   0
C0106    GT0   C110
C0107    360
C0108    X<>Y
C0109    1.
C0110    15
C0111    /
C0112    RCL+   R
C0113    0.06571
C0114    RCLx   T
C0115    1.
C0116    6.622
C0117    1.
C0118    24
C0119    RMDR
C0120    RCL-   G
C0121    RCL+   U
C0122    ->HMS
C0123    RTN

```

===== Date routines for HP33S and HP35S

There are two routines included here.

LBL X takes a date in the form of MM.DDYYYY and returns the day number for the year.

LBL Y takes two dates (MM.DDYYYY in X and Y and returns the number of days between them.

```

X0001    LBL   X   Calculate day of year given date
X0002    ENTER
X0003    ENTER
X0004    100      Get first date by year - 1 day
X0005    x
X0006    FP
X0007    0.0001
X0008    1.
X0009    100
X0010    /
X0011    12.31
X0012    +
X0013    X<>Y
Y0001    LBL   Y   Given two dates (on stack) get days between
Y0002    STO   Z   Save second date
Y0003    X<>Y
Y0004    XEQ   Z   Get day number for first date
Y0005    STO   A   Save day number for first date
Y0006    RCL   Z   Get second date
Y0007    XEQ   Z   Get day number for second date
Y0008    RCL-   A   Subtract day number 1 from day number 2
Y0009    RTN
Z0001    LBL   Z   Calculate day number
Z0002    ENTER
Z0003    IP
Z0004    9
Z0005    +
Z0006    12
Z0007    RMDR    (MOD)
Z0008    STO   M   Save month value
Z0009    X<>Y
Z0010    ENTER
Z0011    FP
Z0012    100
Z0013    x
Z0014    IP
Z0015    STO   D   Save day value
Z0016    RDN
Z0017    100
Z0018    x
Z0019    FP
Z0020    10,000
Z0021    x
Z0022    RCL   M   Get month
Z0023    10
Z0024    INT/
Z0025    1.
Z0026    STO   Y   Save year value
Z0027    365

```

```
Z0028      x
Z0029      RCL   Y   Get year value
Z0030      4
Z0031      INT/
Z0032      +
Z0033      RCL   Y
Z0034      100
Z0035      INT/
Z0036      1.
Z0037      RCL   Y
Z0038      400
Z0039      INT/
Z0040      +
Z0041      RCL   M   Get month value
Z0042      306
Z0043      x
Z0044      5
Z0045      +
Z0046      10
Z0047      INT/
Z0048      +
Z0049      RCL   D   Get day value
Z0050      1
Z0051      1.
Z0052      +
Z0053      RTN
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Incomplete Gamma Function for wp34s

Posted by [Les Wright](#) on 5 Dec 2011, 2:40 a.m.

Most readers will know that the "complete" gamma function is related to the factorial function by the following: $\Gamma(a) = (a-1)!$. For real numbers, this function is computed by integrating $t^{a-1} \exp(-t)$ from $t = 0$ to $t = +\infty$.

The so-called incomplete gamma functions are what one gets when one computes this integration with one of the limits of integration changed to something different. For example, the left-sided incomplete gamma, which here is computed by IGS, is given when one takes the integration from $t = 0$ to $t = \text{some } x$. Analogously the right-sided incomplete gamma function, which here is computed by IGL, takes the integration from $t = \text{some } x$ to $t = +\infty$. Obviously, the sum of IGS and IGL for the same underlying "a" gives the "complete" gamma for that "a".

The left-sided incomplete gamma is best computed by a series approximation when $x < a$, so the routine is call IGS, where the "S" stands for "series". The right-sided incomplete gamma is best computed by a continued fraction expansion using the modified Lentz method when $x > a$, so is called IGL, where the "L" is for Lentz.

The code below was adapted from the routines given in the Numerical Recipes books:

```
001 LBL 'IGS'
002 STO 00
003 X[<->] Y
004 STO 01
005 STO 02
006 1/X
007 STO 03
008 STO 04
009 LBL 00
010 STO 05
011 INC 02
012 RCL 00
013 STO[times] 03
014 RCL 02
015 STO/ 03
016 RCL 03
017 STO+ 04
018 RCL 05
019 RCL 04
020 X[!=]? Y
021 GT0 00
022 GT0 04
023 LBL 'IGL'
024 STO 00
025 X[<->] Y
026 STO 01
027 -
028 1
029 +
030 STO 02
031 1
032 EEX
033 2
034 5
035 0
036 STO 05
```



```

037 1/x
038 STO 06
039 +
040 1/x
041 STO 03
042 STO 04
043 CLX
044 STO 08
045 LBL 09
046 INC 08
047 INC 02
048 INC 02
049 RCL 01
050 RCL - 08
051 RCL[times] 08
052 STO 07
053 RCL[times] 03
054 RCL+ 02
055 RCL+ 06
056 STO 03
057 RCL 07
058 RCL/ 05
059 RCL+ 02
060 RCL+ 06
061 STO 05
062 RCL 03
063 STO/ 03
064 STO/ 03
065 /
066 STO[times] 04
067 x[!=]1?
068 GT0 09
069 RCL 04
070 LBL 04
071 RCL 00
072 RCL 01
073 y[^x]
074 [times]
075 RCL 00
076 +/-
077 e[^x]
078 [times]
079 RTN

```

Use is simple: put the arguments on the stack as a [ENTER] x and XEQ either IGS or IGL. Given the high internal accuracy of the wp34s, the displayed digits (and then some) are almost always completely accurate.

Example: Compute the left and right incomplete gammas for a = 5 and x =5.

5 [ENTER] 5 {XEQ} [alpha] IGS returns 13.4281611584 in ALL 00.

5 [ENTER] 5 {XEQ} [alpha] IGL returns 10.5718388416.

It can be easily seen that these outputs sum to 24, which is the "complete" gamma of 5, or the factorial of 5-1 = 4.

I am aware that this routine may be of only academic interest to most. The incomplete gamma functions are of practical use in that they form the basis of the error function, the cumulative standard normal distribution, and the cumulative chi-square distribution. All of these applications have their own excellent routines already built into the wp34s. I originally ported the Numerical Recipes program for use on the HP41C, HP42S, HP35s, and older calcs that did not have these statistical functions built in. Nonetheless, I hope this code interests someone.

Edited: 5 Dec 2011, 6:04 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

Romberg Integration on the wp34s

Posted by [Les Wright](#) on 4 Dec 2011, 7:43 p.m.

Romberg's Method of numerical integration entails iteratively applying a simple quadrature scheme such as the trapezoidal rule or midpoint (rectangle) rule by adding more function evaluations at each pass through the algorithm and accelerating convergence using the century-old method of Richardson extrapolation. The underlying mathematics are not terribly complicated, and I can recommend the Wikipedia article on the subject. (However, one should keep in mind that that article's treatment builds the algorithm around the trapezoidal rule, whereas what follows uses the rectangle rule. But the principles are the same.)

An adapted version of Romberg's Method underlies the numerical integration abilities of the HP34C, HP15C, HP33s, HP35s, and the INTEG routine of the HP41C Advantage Pac. As of version 2.2, it is NOT used by the wp34s, which instead uses a very fast and good 10/21 Gauss-Kronrod scheme. However, some quadrature aficionados may prefer a small and relatively fast alternative on the wp34s that performs more like the built-in integrator on the above calculators. I can take no credit for the underlying programming, but I am glad to provide the following port.

The following is a nearly keystroke-by-keystroke translation of the HP41C PPC ROM Romberg integrator, IG. I have long been fascinated by the economy of this routine since discovering it in the PPC ROM manual on the MOHPC DVD set. The earliest HP implementation of Romberg's Method was in the 34C by William Kahan. In keeping with Kahan's work, IG just doesn't apply Romberg directly to the function. There are first two transformations of variable--the first a linear one to standardize the interval of integration to $[-1,1]$, the second a nonlinear one so that points are sampled at uneven intervals and problems with some periodic functions are avoided. It is at this point that I put in an unapologetic plug for the MOHPC DVD--the PPC ROM manual and the August 1980 HP Journal article describing the 34C integration capacities are, for me, in themselves well worth the cost.

Without ado, here is the routine. It is in proper syntax and can be simply cut, pasted, and assembled for use in the emulator and transferred to the calculator:

```
001 LBL 'IG'
002 STO 17
003 X[<->] Y
004 -
005 4
006 /
007 STO 16
008 STO- 17
009 STO- 17
010 0
011 STO 15
012 STO 11
013 STO 18
014 SF 09
015 LBL 01
016 1
017 ENTER[^]
018 2
019 STO 14
020 RCL 11
021 +/-
022 y[^x]
```

```

023 STO[times] 14
024 1
025 -
026 LBL 02
027 STO 12
028 x[^2]
029 -
030 STO 13
031 2
032 +
033 RCL[times] 12
034 RCL[times] 16
035 RCL+ 17
036 XEQ[alpha]
037 RCL[times] 13
038 STO+ 15
039 1
040 RCL 12
041 RCL+ 14
042 x<? Y
043 GTO 02
044 RCL 11
045 STO 13
046 1
047 8
048 STO 12
049 1
050 STO+ 11
051 RCL 15
052 RCL 16
053 1
054 .
055 5
056 [times]
057 [times]
058 RCL[times] 14
059 LBL 03
060 R[^]
061 4
062 [times]
063 ENTER[^]
064 DSE Y
065 x[<->] Z
066 ENTER[^]
067 x[<->][>-]12
068 STO- Y
069 ROUND
070 x[<->] Z
071 /
072 RCL+[>-]12
073 INC 12
074 DSE 13
075 GTO 03
076 STO[>-]12
077 FS? 10
078 VIEW X
079 FS?C 09
080 GTO 01
081 ROUND
082 x[!=]? Y
083 GTO 01
084 RCL L
085 RTN

```

The program uses the 4-stack and R11 through R[>-]12. As we can see in the Wikipedia article (and the PPC ROM article for those who have it), Romberg's Method produces a lower triangular matrix with the best estimate after a given iteration located in the rightmost cell of the last row. In IG, only the most recent row is saved in R18 to R[>-]12. The routine continues until the rightmost elements of the two most recent rows agree to the desired number of digits. (Note that in the Wikipedia article a different convergence criterion is used--i.e., the last two elements of the last row agree. This is less generally stringent.)

Since comparison of rounded values determines when IG stops, the setting of FIX, SCI, ENG or ALL determines accuracy, as it does in old HP calcs (except for the 42S, where an actual number for accuracy was entered). In general, SCI nn gives more predictable results--we know, for example, that SCI 04 will give us five significant digits, whereas with FIX nn that can vary according to the number of leading zeros. ALL nn gives the same results as SCI 11 for any

nn.

As in the original IG, if flag 10 is set interim estimates are displayed until convergence is achieved. I like keeping this set so I can watch the convergence happen. I can tell things are done when the little "RCL" annunciator stops flashing. But this port differs from the original IG in that the global alpha label of the routine to be integrated is stored in the alpha register, not R10. IG as it is presently written does not integrate routines labeled by numeric or local alpha labels (unless a globally labeled wrapper is provided). This behaviour makes it similar to the Advantage Pac's INTEG.

Here are instructions for use:

1. Enter the function to be integrated. It must be labeled by a global alpha label, end with RTN, not use R11 through to about R30 (rarely IG will need more for the last row of the Romberg matrix), and return the function value to the X register.
2. Enter the alpha name of this function in the alpha register.
3. Set flag 10 if you want to see interim results.
4. Set the desired accuracy through the display settings--SCI nn is recommended to give nn+1 significant digits.
5. Put the limits of integration on the stack: lower [ENTER] upper.
6. Execute the routine ([XEQ] [ENTER] IG [ENTER]) and watch the magic unfold.

Here is an example given in the 1980 Kahan article: **integrate $2*w^2/((w-1)*(w+1)) - w/\ln(w)$ from $w = 0$ to $w = 1$.**

I enter the following routine:

```
001 LBL 'KAH'
002 FILL
003 1
004 -
005 x[<->] Y
006 1
007 +
008 [times]
009 /
010 ENTER[^]
011 +
012 x[<->] Y
013 LN
014 1/x
015 -
016 [times]
017 RTN
```

```
SF 10 0 [ENTER] 1 [f] [alpha] KAH [ENTER] [SCI] 11 [XEQ] [ENTER] IG [ENTER]
```

eventually gives 3.64899739786E-2. This is indeed the correct result to 12 digits. It took about 80 seconds on my wp34s, but I should note my calc is clocked down to half speed due to battery drain. When the batteries were fresher, this integral took at most 45 seconds. Of course, it is much faster if the desired digits are set lower.

Of note, SHOW tells us that the internal 16-digit result is 3.648997397857771e-2. According to Mathematica, the integral does indeed have an exact result--it is $2 - \text{Euler's gamma} - \ln(4)$ --which is 3.648997397857652e-2 to 16 digits. This is remarkably close agreement.

I close with a couple of notes. First of all, there is nothing particularly sacred about using R11 and up, and that can be changed. Second, some users may choose to modify the code so that no global alpha labelling of the function routine is necessary. Finally, in contrast with the output of the 34C, 35s, and 15C, we get no error estimate explicitly with this routine. Personally, I have been using R[->]12 - R18, the difference between the final result and the most recent unextrapolated rectangle estimate (stored as the first element in the the most recent row for the Romberg tableau) as a

rough upper limit on the absolute error.

Reference:

[link:http://en.wikipedia.org/wiki/Romberg's_method]Wikipedia Article on Romberg Integration[/link]

Edited: 6 Dec 2011, 1:04 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

WP 34S: Polynomial root finder adapted from Valentin Albillo's HP 41 program

Posted by [Miguel Toro](#) on 23 Nov 2011, 7:32 p.m.

Usage:

1. [XEQ][ENTER]'PRT'
2. Enter degree [R/S]
3. Enter coefficients: Imaginary part [ENTER] real part [R/S]
note : if only real, enter the value and [R/S]

Resources:

Flags
A, 01

Registers
A: iterations limit
r09-rNx2+10: coefficients and roots
r00: root counter
r01: index to coefficients / solutions
r03-r04: auxiliary registers
r05: index to the coefficients
r07-r08: auxiliary registers

Labels
00,01,02,05,06,10,11,12

Comments:
Works with WP34s ver. 3.2

Program:

```
001 LBL 'PRT'
002 SSIZE4
003 FIX 00
004 CL[alpha]
005 [alpha]'N=?'
006 PROMPT
007 STO 00
008 STO 03
009 STO+ X
010 9
011 .
012 0
013 0
014 8
015 0
016 2
017 +
018 STO 01
019 STO 05
```

```

020 CLX
021 LBL 05
022 CL[alpha]
023 [alpha]'Im[^]'
024 [alpha]'Re[space]'
025 [alpha]IP 03
026 [alpha]?
027 CLSTK
028 PROMPT
029 [cmp]x]STO[->]05
030 DEC 03
031 DSE 05
032 GTO 05
033 RCL 03
034 LBL 06
035 4
036 SDL 002
037 STO A
038 x[<->] Y
039 CF A
040 +/-
041 STO 04
042 FIX 02
043 ROUND
044 FIX 11
045 x[!=]0?
046 GTO 01
047 1
048 STO 04
049 LBL 01
050 DSE A
051 SKIP 01
052 GTO 12
053 RCL 00
054 STO 08
055 SF 01
056 XEQ 11
057 [->]POL
058 1/x
059 STO 07
060 x[<->] Y
061 +/-
062 STO 08
063 CF 01
064 XEQ 11
065 [cmp]x]RCL 07
066 [->]REC
067 [cmp]x][times]
068 [cmp]x]STO- 03
069 [cmp]x]ROUND
070 [cmp]x]x[!=]0?
071 GTO 01
072 SF A
073 XEQ 11
074 2
075 STO+ 05
076 SDR 003
077 STO+ 01
078 [cmp]x]RCL 03
079 [cmp]x]STO[->]05
080 x[<->] Y
081 DSE 00
082 GTO 06
083 RCL 01
084 IP
085 # 10
086 -
087 SDR 003
088 STO- 05
089 FIX 07
090 LBL 10
091 INC 00
092 CL[alpha]
093 [alpha]'Re[space]'
094 [alpha]IP 00
095 [cmp]x]RCL[->]05
096 x[approx]0?
097 CLSTK
098 PROMPT
099 x[<->] Y
100 x[approx]0?

```

```
101 SKIP 04
102 CL[alpha]
103 [alpha]'Im[space]'
104 [alpha]IP 00
105 PROMPT
106 DSE 05
107 GTO 10
108 CLSTK
109 CF A
110 RTN
111 LBL 11
112 RCL 01
113 STO 05
114 FC? 01
115 GTO 13
116 EEX
117 3
118 +/-
119 ENTER[^]
120 2
121 [times]
122 STO+ 05
123 LBL 13
124 [cplx]RCL[->]05
125 FC? 01
126 GTO 02
127 RCL 08
128 STO[times] Z
129 [times]
130 DSE 08
131 GTO 02
132 RTN
133 LBL 12
134 CL[alpha]
135 [alpha]'No[space]'
136 [alpha]'Os'
137 VIEW[alpha]
138 CLSTK
139 RTN
140 LBL 00
141 [cplx]RCL 03
142 [cplx][times]
143 RCL[->]05
144 FS? 01
145 RCL 08
146 FS? 01
147 [times]
148 +
149 FS? A
150 STO[->]05
151 x[<->] Y
152 INC 05
153 RCL[->]05
154 FS? 01
155 RCL 08
156 FS? 01
157 [times]
158 +
159 FS? A
160 STO[->]05
161 DEC 05
162 x[<->] Y
163 FS? 01
164 DSE 08
165 LBL 02
166 DSE 05
167 GTO 00
168 RTN
169 END
```

Edited: 30 Aug 2013, 3:54 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

WP 34S: financial routines IRR, NPV, Amortization, Interest rate conversion (updated 2013-04-28)

Posted by [Miguel Toro](#) on 12 Nov 2011, 2:20 p.m.

/* Finance Routines

What is included in this file:

1. Cash flow Analysis
(Net present value and Internal rate of return)
2. Amortization
3. Nominal vs. Effective interest rate conversion

How to use:

1. Copy all this text in a new file called something like "fin.wp34s", not forgetting the comment marks.

2. Assemble the file. Here is an example on Windows:

```
wp34s_lib -ilib wp34s.dat -cat -state -olib wp34s.dat fin.wp34s
```

3. Transfer the .dat file to the emulator and/or your calculator.

Notes:

- The routines can be executed from flash.
- These are been tested on version 3.2 revision 3376

1. CASH FLOW ANALYSIS -----

a.- CFE - Cash Flow input

- Enters a maximum of 49 cash flow pairs for NPV and IRR calculation
- Usage:
Call the program and enter each pair of cash flow and occurrences and finish entering with 0 and [R/S]. A second [R/S] executes the Cash Flow Review routine for easily verify the entries.

Input	Alpha	x
-----	-----	-----
[XEQ], [ENTER], CFE	CF0^n0	0.
.	CF#^n#	0.
CF#, [ENTER], n#, [R/S]		
-- or --		
0, [R/S]	#CF=	# of cash flows
[R/S] ---> see CFR		

b. CFR - Cash Flow Review

- Review and modification of an existing cash flows series
- Usage:
Call the program and browse through the cash flow pairs. When a value is shown it can be modified before continuing the review, pressing the [R/S] key. This is a circular routine.

Input	Alpha	x
[XEQ], [ENTER], CFR	CF0=	CF0 value
.	CF#=#	CF# value
(new cf - optional), [R/S]	n#=#	n# value
(new n - optional), [R/S]	...	

c. NPV - Net Present Value

- Calls NPV routine (LBL 00) to calculate NPV values
- Usage:
Input the Interest rate per period and execute NPV to calculate the Net Present Value based on previously entered Cash Flows(see CFE). The result will be shown in register x and the interest rate per period is kept in register I.

Input	Alpha	x
I/P%		I/P% value
[XEQ], [ENTER], NPV	NPV=	NPV value

Additional Output :
I: I/P% for further usage

d. IRR - Internal Rate of Return

- Solves NPV routine (LBL 00) to find an IRR value
- Usage:
Call IRR based on a previously entered Cash Flows (see CFE).

Input	Alpha	x
[XEQ], [ENTER], IRR	IRR=	IRR value

2. AMORTIZATION -----

a. ADE - Amortization Data Entry

- Obtains the required data to calculate an amortization schedule: Capital, Payment per period, Number of periods and Interest rate per period (see the WP34s TVM application).
- Usage:
Call the routine and provide the following data:
 - Capital: Positive value,
 - Payment: negative value,
 - Total periods: if information is in years, multiply it by the number of periods per year,
 - Interest rate: if annual interest, divide it by the number of periods per year.

Input	Alpha	x
[XEQ], [ENTER], ADE	Cap?	previous Cap value if any
Cap value or nothing, [R/S]	Pmt?	previous Pmt value if any
Pmt value or nothing, [R/S]	Tprds?	previous total of periods if any
Tprds value or nothing, [R/S]	I/P%?	previous Interest/period if any
I/P% value or nothing, [R/S]		0.

b. AMT - Amortization calculation

- Calculates an amortization schedule for a interval of time based on data previously entered with ADE
- Usage:
Call AMT and enter an initial period and a final period. if you want to calculate for just a period (ex: a month), just enter the initial period and press [R/S] when asked for the final period. the routine finishes, entering 0 when asked for a new initial period.

Input	Alpha	x
[XEQ], [ENTER], AMT	Prd I?	0.
Prd I, [R/S]	Prd F?	Prd I
Prd F, [R/S]	Pri	principal amount
[R/S]	Int	interest amount

[R/S]	Bal	Balance at period F
[R/S]	Prd I?	0.
0, [R/S] ---> ends routine		

3. NOMINAL/EFFECTIVE INTEREST RATE CONVERSION -----

a. EFF - Effective interest rate calculation

Calculate the effective interest rate base on the compounding ...

- Usage:

Input the Interest rate per period and execute NPV to calculate the Net Present Value based on a previously entered Cash Flow series using CFE. The result will be shown in register x and the interest rate per period is kept in register I.

Input	Alpha	x
-----	----	-----
Nominal %, [XEQ], [ENTER], EFF	P/Y?	Previous P/Y if any
new P/Y or nothing, [R/S]		Effective %

b. NOM - Nominal interest rate calculation

Calculate the Nominal interest rate base on the compounding ...

- Usage:

Input the Interest rate per period and execute NPV to calculate the Net Present Value based on a previously entered Cash Flow series using CFE. The result will be shown in register x and the interest rate per period is kept in register I.

Input	Alpha	x
-----	----	-----
Effective %, [XEQ], [ENTER], EFF	P/Y?	Previous P/Y if any
new P/Y or nothing, [R/S]		Nominal %

*/

/* 1. Cash Flow Analysis */

/* Cash Flow input - CFE

- Enters a maximum of 49 cash flow pairs for NPV and IRR calculation

- Usage:

Call the program and enter each pair of cash flow and occurrences and finish entering with 0 and [R/S]. A second [R/S] executes the Cash Flow Review routine for easily verify the entries.

Input	Alpha	x
-----	----	-----
[XEQ], CFE	CF0^n0	0.
.		
CF#, [ENTER], n#, [R/S]	CF#^n#	0.
-- or --		
0, [R/S]	#CF=	# of cash flows
[R/S] ---> see CFR		

*/

```

LBL'CFE'
CLSTK
STO I
STO J
STO K
cfeloop::
CLx
CL[alpha]
"CF"
[alpha]IP J
"[^]n"
[alpha]IP J
PROMPT
x=0?
JMP endloop
STO+ I
[cmplx]STO[->]K

```

```

                                INC J
                                INC K
                                INC K
                                JMP cfeloop
endloop::
                                RCL K
                                x=0?
                                ERR 15
                                2
                                -
                                SDR 003
                                2
                                SDR 005
                                +
                                STO K
                                CLx
                                CL[alpha]
                                [alpha] #
                                "CF="
                                VW[alpha]+ I
                                STOP

/* Cash Flow Review - CFR
- Review and modification of an existing cash flows series
- Usage:
  Call the program and browse through the cash flow pairs. When a value
  is shown it can be modified before continuing the review, pressing
  the [R/S] key. This is a circular routine.
  Input          Alpha          x
  -----
  [XEQ], [ENTER], CFR          CF0=          CF0 value
  .
  (new cf - optional), [R/S]    CF#=          CF# value
  (new n - optional), [R/S]    n#=          n# value
  ...
*/
                                LBL 'CFR'
                                RCL K
                                STO J
                                STO I
cfrloop::
                                CL[alpha]
                                "CF "
                                [alpha]IP I
                                [alpha] =
                                INC J
                                RCL[->]J
                                PROMPT
                                STO[->]J
                                CL[alpha]
                                [alpha] n
                                [alpha]IP I
                                [alpha] =
                                DEC J
                                RCL[->]J
                                PROMPT
                                STO[->]J
                                INC I
                                ISG J
                                JMP cfrloop
                                GTO 'CFR'

/* Net Present Value
- Calls NPV routine (LBL 00) to calculate NPV values
- Usage:
  Input the Interest rate per period and execute NPV to calculate the
  Net Present Value based on previously entered Cash Flows(see CFE).
  The result will be shown in register x and the interest rate per
  period is kept in register I.

  Input          Alpha          x
  -----
  I/P%
  [XEQ], [ENTER], NPV          NPV=          NPV value

Additional Output :
  I: I/P% for further usage
*/
```

```
LBL'NPV'  
STOM D  
SSIZE4  
XEQ npv_irr  
FIX 02  
CL[alpha]  
"NPV="   
VW[alpha]+ X  
EEX  
2  
STO[times] I  
R[v]  
RCLM D  
RTN
```

/* Internal Rate of Return
- Solves NPV routine (LBL 00) to find an IRR value
- Usage:
 Call IRR based on a previously entered Cash Flows (see CFE).

Input	Alpha	x
-----	-----	-----
[XEQ], [ENTER], IRR	IRR=	IRR value

```
*/  
  
LBL'IRR'  
STOM D  
SSIZE4  
FIX 06  
CLSTK  
EEX  
2  
SLV npv_irr  
JMP irrfound  
x[approx]? Y  
JMP irrfound  
RCL Z  
FIX 02  
ROUND  
x[!=]0?  
ERR 20  
R[v]  
irrfound::  
FIX 02  
CL[alpha]  
"IRR="   
VW[alpha]+ X  
RCLM D  
npv_irr::  
RTN  
1 // Calculates NPV and solves for IRR  
STO A  
%  
STO I  
STO B  
INC B  
CLX  
STO C  
RCL K  
STO J  
sigmanpv::  
[cmp]x]RCL[->]J  
RCL I  
x=0?  
JMP intzero  
LN1+x  
+/-  
[times]  
e[^x]-1  
STO A  
INC A  
+/-  
RCL/ I  
RCL[times] B  
ENTER[^]  
intzero::  
R[v]  
[times]  
STO+ C  
RCL A  
STO[times] B
```

```

ISG J
JMP sigmanpv
RCL C
RTN

```

/* 2. Amortization */

/* Amortization Data Entry - ADE

- Obtains the required data to calculate an amortization schedule: Capital, Payment per period, Number of periods and Interest rate per period (see the WP34s TVM application).

- Usage:

Call the routine and provide the following data:

- Capital: Positive value,
- Payment: negative value,
- Total periods: if information is in years, multiply it by the number of periods per year,
- Interest rate: if annual interest, divide it by the number of periods per year.

Input	Alpha	x
-----	-----	-----
[XEQ], [ENTER], ADE	Cap?	previous Cap value if any
Cap value or nothing, [R/S]	Pmt?	previous Pmt value if any
Pmt value or nothing, [R/S]	Tprds?	previous total of periods if any
Tprds value or nothing, [R/S]	I/P%?	previous Interest/period if any
I/P% value or nothing, [R/S]		0.

*/

```

LBL 'ADE'
SSIZE4
CL[alpha]
RCL A
"Cap?"
PROMPT
STO A
RCL B
CL[alpha]
"Pmt?"
PROMPT
STO B
RCL C
CL[alpha]
"Tprds?"
PROMPT
STO C
RCL I
SDL 002
CL[alpha]
"I/P%?"
PROMPT
SDR 002
STO I
CLX
RTN

```

/* Amortization calculation - AMT

- Calculates an amortization schedule for a interval of time based on data previously entered with ADE

- Usage:

Call AMT and enter an initial period and a final period. if you want to calculate for just a period (ex: a month), just enter the initial period and press [R/S] when asked for the final period. the routine finishes, entering 0 when asked for a new initial period.

Input	Alpha	x
-----	-----	-----
[XEQ], [ENTER], AMT	Prd I?	0.
Prd I, [R/S]	Prd F?	Prd I
Prd F, [R/S]	Pri	principal amount
[R/S]	Int	interest amount
[R/S]	Bal	Balance at period F
[R/S]	Prd I?	0.
0, [R/S] ---> ends routine		

*/

```

LBL 'AMT'
SSIZE4
CLX

```

```
CL[alpha]
"Prd-I?"
PROMPT
x=0?
JMP finamt
ENTER[^]
CL[alpha]
"Prd-F?"
PROMPT
[cmplx]STO Z
x[<->] Y      // Paiments for period F-I
-
INC X
RCL[times] B
STO D
R[v]
XEQ princip    // Principal for F
x[<->] Y
DEC X
XEQ princip    // Principal for I-1
-              // Principal for period F,I
CL[alpha]
"Pri"
PROMPT
STO- D         // Interest for period F,I
RCL D
CL[alpha]
"Int"
PROMPT
RCL A          // Balance = Capital - Principal for F
RCL+ T
CL[alpha]
"Bal"
PROMPT
GTO'AMT'
finamt::
princip::
XEQ accfact
RCL C
XEQ accfact
/
RCL[times] A
+/-
RTN
accfact::
RCL I
LN1+x
[times]
e[^x]-1
RTN

/* 3. Nominal/Effective interest rate conversion */
/* Effective interest rate calculation - EFF
Calculate the effective interest rate base on the compounding ...
- Usage:
Input the Interest rate per period and execute NPV to calculate the
Net Present Value based on a previously entered Cash Flow series using
CFE. The result will be shown in register x and the interest rate per
period is kept in register I.

Input          Alpha          x
-----
Nominal %, [XEQ], [ENTER], EFF    P/Y?    Previous P/Y if any
new P/Y or nothing, [R/S]          Effective %

*/
LBL'EFF'
SF C
JMP prep

/* Nominal interest rate calculation - NOM
Calculate the Nominal interest rate base on the compounding ...
- Usage:
Input the Interest rate per period and execute NPV to calculate the
Net Present Value based on a previously entered Cash Flow series using
CFE. The result will be shown in register x and the interest rate per
period is kept in register I.

Input          Alpha          x
```

Effective %, [XEQ], [ENTER], EFF
new P/Y or nothing, [R/S]

P/Y?

Previous P/Y if any
Nominal %

```
*/
LBL 'NOM'
CF C
prep:: STOM J
SSIZE4
periods:: x[<->] D
CL[alpha]
"p-y?"
PROMPT
x[<->] D
FS? C
calcnom:: JMP calceff
STO I
0
ENTER[^]
EEX
2
SLV nom_eff
JMP nomfound
FIX 06
x[approx]? Y
JMP nomfound
FIX 02
x[<->] Z
ROUND
x[!=]0?
ERR 20
nomfound:: x[<->] Z
STO I
RCLM J
RTN
calceff:: 0
STO I
R[v]
XEQ nom_eff
STO I
RCLM J
RTN
nom_eff:: RCL D // Calculates EFF and solves for NOM
SDL 002
/
1
+
RCL D
y[^x]
1
-
SDL 002
RCL - I
RTN
END
```

Edited: 29 Apr 2013, 1:11 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Inside Saturn Emulator Timing

Posted by [Christoph Giesselink](#) on 20 Oct 2011, 4:56 p.m.

In the last time we read more and more about emulators, especially since HP use this technique to bring old calculator designs back to life in real hardware (12C and 15C of the Voyager series).

But writing an emulator isn't a trivial task. On one side you want to have the accuracy in behavior of the original machine, on the other side you want to use the benefits of the higher speed of the host platform. It's something like to want to have the "quadrature of the circle". This article want to show some of the consequences in the emulation inside Emu28, Emu42 and Emu48.

1) Real Saturn hardware

The following information should be understand as an overview over the Saturn family and not as feature included in every hardware design.

All machines use a 32768Hz crystal for the main clock.

- The 32768Hz crystal clock is divided by 4 to get the 8192Hz timer clock.
- The 32768Hz crystal clock is divided by 8 to get the 4096Hz clock of the display line refresh in the Clarke and Yorke chip
- The 8192Hz timer clock is divided by 8 to get the 1024Hz (1ms) keyboard timer
- The 8192Hz timer clock is divided by 512 to get the 16Hz timer

So all these timers have the accuracy of the original 32768Hz crystal.

The CPU clock (1MHz at Lewis chip) is generated by a PLL from the 32768Hz crystal. But this frequency multiplier isn't very accurate. The resulting frequency depends on temperature, humidity and other ambient influences.

2) Measuring the CPU clock speed

In some cases the software should know the real CPU clock speed. In the case of the HP42S for example for generating the Redeye infrared signal for the HP82240 printer.

How speed measurement is done?

In a typical 100m sprint race you have a fixed distance and measure the time. So the speed is the 100m divided by the elapsed time. Another possibility is setting a fixed time and measure the distance. Which measurement method you should use depends on the accuracy of each input. If you have an better accuracy in time measurement, you measure the time, else the distance.

As you know CPU speed is defined as "CPU ticks / time". In our case we have more CPU speed ticks in one time unit,

so we should use a fixed time and measure the CPU speed ticks.

So the speed measuring pseudo code is quite easy:

1. preset a loop counter to 0
2. read timer and wait until timer content change
3. subtract from last timer reading the timer ticks for the wanted measuring time
4. increment loop counter
5. read actual timer value and compare it with the calculated end timer value
6. until end timer value not reached goto 4.

Step 2. makes sure that you always measure complete timer ticks.

So the clock speed is "loop counter" * "CPU cycles per loop" / "measure time".

3) The Emu28, Emu42 and Emu48 emulators

As I wrote in the 1st section an emulator should fulfill the "quadrature of the circle". Maximum speed at all (mathematical) calculations, correct timer references and a fully working user interface (UI).

Because we are speaking about an emulator no ROM image changes should be necessary. But an emulator cannot distinguish between Math calculations and running a game for example. So the above emulators have a method to slow down CPU speed at some situations. The easiest situation is the user tells, that the calculator should run at normal speed. But more critical is full speed. The emulators today are 10 to 100 or more times faster than the original calculator hardware. One problem for example is the auto-repeat function of the backspace key at the HP48. The delay for the key auto repeat is completely given by CPU cycles. So a CPU cycle operating delay of 200ms will purge 5 characters/second. If now the CPU is 100 times faster, we purge 500 characters/second. So without any further work on the emulator the usability of the UI at an emulated HP48 was between "limited to unusable". My solution for the auto repeat: When the emulator know that a key is pressed, it switch down to normal speed, when all keys are released the emulator return to full speed.

Next problem is the speed measurement. Normally it would be fine that the speed measurement will return the correct speed, and all functions which are using this return value will operate properly with the much higher value. But we live in a real world limited by the resources of the real hardware and many other boundaries. In the case of speed measurement the word size of the loop counter is to small, so the loop counter overrun several times and will return any random number. And with this random number the HP42S and other calculators try to calculate the timing for the redeye printer interface. But the timing calculation routine only works properly with speed values in a reasonable range given by the original hardware.

This limitation took me ~7 years to add infrared printing to Emu42 without a ROM modification. Ok, I don't worked 7 years on it, over 6 years the project was canceled, because I had no suitable idea how to solve the speed measurement problem.

Solution: Because I cannot modify the speed measurement routine in the firmware, I have to slow down emulation speed during the measurement to get proper values without a loop counter overflow. Therefore I check the CPU cycles at every 8192Hz timer read access. Are the measured CPU cycles below 150, I'm sure that the program reads the timer value in a very short interval, maybe in loop. This is a typical timer read access pattern at speed measurement. In this case I also slow down emulation to normal speed. To switch back to fast speed I should have something like a retriggerable mono-flop. Each time I recognize the slow down condition I retrigger the mono flop. Has mono flop time gone, the emulation speed can be switched back to fast speed. The mono flop is realized as an opcode counter in my emulators. On the trigger condition I tell the system, to slow down emulation speed for the next 10 opcodes. So the emulator has only to keep track on the remaining number of opcodes to slow down. A very simple solution with less overhead.

4) Conclusion

Making an emulator isn't just rebuild the hardware as close as possible even more when you want to use the benefits of a more powerful host system. There's another big difference to HP and my emulator development. As a private project I don't have to hurry and have the time to always think about a "best" solution. In commercial development, with respect of time and cost, not the "best", but rather an "optimal" solution is requested.

Christoph Giesselink c dot giesselink at gmx dot de

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 15 LE: Errata list for Owner's Handbook

Posted by [Felix Gross](#) on 10 Oct 2011, 3:03 a.m.

Here is a list of errors I saw reported in the Forum or I found otherwise. The first source known to me is indicated. Thanks to all eagle-eyed proof readers.

As a non-native English speaker I am probably not the right person to decide on style, so mere suggestions on improving the style of the LE Manual are not included.

Please continue to report errors. I will update the list regularly.

Please note that there is an updated Sep 2011 Edition 2.4 to the printed Owner's Handbook on HP's calculator site.

1. Page 20: Note: Decimal digits from the mantissa that spill into the exponent field will disappear from the display when you press ", but will be retained internally. The stray quotation marks are the problem. An EEX should be inserted instead of the stray quotation mark. (Marcus von Cube)
2. Page 22: Paragraph under the section headline "Two-Number Functions and ENTER": In the last sentence there should be a comma before the "and". (Rich Messeder)
3. Page 23: Last equation is in the wrong typeface. (Felix Gross)
4. Page 34: Last line in the last paragraph. It should be one Roll-up and one Roll-down, not two Roll-ups. The figure under the paragraph is correct. (Georg Becker)
5. On page 35, bottom of the page, the [+] should be a [$\div$] in two instances. (troy)
6. Page 41: Third paragraph: "If the variables change (as in the preceding example), be sure and clear the display before entering the new variable" should most likely be "If the variables change (as in the preceding example), be sure to clear the display before entering the new variable. The LE Manual is the same as the original Manual (Rich Messeder)
7. On page 42 of the second sentence of the first paragraph reads in part Quote: ... (see the diagram of the registers on the inside back cover). There is no register diagram on the inside back cover. (Howard Owen)
8. Page 53, second paragraph: The punctuation around the x and y is inconsistent as in the original. (Rich Messeder)
9. Page 57, second paragraph: In the symbol set following "Polar vector coordinates must be converted to rectangular coordinates upon entry" there should be a comma after the "r". This is also missing in the original manual. (Rich Messeder)
10. Page 57, second paragraph, sixth line: the hyphen is really a minus sign of the number "200". (Rich Messeder)
11. Page 61, changing the decimal separator from a period to a comma the example shows: 12.345.6700 instead of the

correct format: 12.345,6700. (Michael de Estrada)

12. Page 62: If the calculator is turned off, Continuous Memory will be preserved for a short period while the batteries are removed. Data and programs are preserved longer than other aspects of calculator status. Refer to appendix F for instructions on changing batteries.

I doubt these facts are true. If you remove both batteries the memory will be cleared. There is no technical reason why the status and data should have a different retention behavior. This was the case when data memory and status were stored in different physical chips on the original hardware but on the Atmel it's all in one chip. (Marcus von Cube)

13. On page 74, last paragraph: It should be "(1,2,...,9,0 from ..." instead of "(1,2,9,0 from ..". (Georg Becker)

14. On page 75 of the printed version: Calculator picture: The highlighted ("TAN") key not too obvious (This is in color in the pdf file). (Thomas Chrapkiewicz)

15. On page 78, second paragraph from the bottom: There should be a "," between the Arrow keystroke and the ON keystroke (Georg Becker)

16. On pages 93, 94: The correct form of the unit Curie is "Ci". Therefore, it should be "mCi" instead of mci. The original manual has mci as well. (Rich Messeder)

17. Page 104: First letter on page should be bold as well. (Michael Kussmaul)

18. Page 105, second last paragraph: There is a spurious character the GSB key. (Rich Messeder)

19. Page 109, at the bottom: Should be "increment or decrement" instead of "of". (Marcus von Cube)

20. Page 110 Top of Page: Arrows/parentheses not aligned properly on nnnnn.xxxyy. Same in .pdf copy. (Thomas Chrapkiewicz)

21. Page 114: In the example the first key stroke sequence runs into the second line. (Rich Messeder)

22. Page 116, footnote: "Refer to pages 58-59" instead of "Refer to page 58-59". (Rich Messeder)

23. On page 124, under Manipulating the Real and Imaginary Stacks, "...twice restore a number to..." should be "...to restore a number to..." (Mike Morrow)

24. Page 131, last paragraph: "The $x \leftrightarrow y$ function. for instance," should be "The $x \leftrightarrow y$ function, for instance," (Rich Messeder)

25. Page 133, last line of the footnote: the comma is spurious. (Rich Messeder)

26. Page 135, first paragraph: "express the result in polar form, (In phasor form," should be "express the result in polar form. (In phasor form," Also, the angle mark in this sentence is misaligned between the 2 and 65° . (Rich Messeder)

27. Page 139: 2 ENTER 1 f DIM B: B is on new line, in original manual, all is on one line. (Michael Kussmaul)

28. Page 140: f MATRIX 0: there are missing spaces before and after the MATRIX key (anyway spacing is not consistent across document). (Michael Kussmaul)

29. Page 148, first footnote: "(possibly with same rows" should be "(possibly with some rows" (Rich Messeder)

30. Page 150: "The Frobenius of Euclidean norm" should be (2 places) "The Frobenius, or Euclidean norm, ". (Rich Messeder)

31. Page 153, second paragraph: The "or" should not be in italics. (Rich Messeder)

32. Page 160: "You don't need to activate Complex mode for calculations with complex matrices." The whole sentence should be in italics. (Rich Messeder)
33. Page 170: 4 ENTER 1 f DIM B: B is on new line, in original manual, all is on one line. (Michael Kussmaul)
34. Page 171: In the equation a space is missing before the "i". (Rich Messeder)
35. Page 173: In the right set of keystrokes, the DIV key is in the wrong line. (Rich Messeder)
36. Page 174, fourth paragraph. After "register" should be a full stop not a comma. (Rich Messeder)
37. Page 176: The colon after the Matrix D should not be in bold. (Rich Messeder)
38. Page 178: In "Forbenius or Eucliedean..." there should be a comma before the "or". The mistake is also in the original. (Rich Messeder)
39. Page 178: In the entry for f MATRIX 9 there should be a full stop after the word "register" not a comma. (Rich Messeder)
40. Page 184: There is a stray "r" near the graph for f(x). (Rich Messeder)
41. Page 186: There is a stray "G" next to the graph second graph. (Felix Gross)
42. Page 186: In the first equation the period should be removed. (Rich Messeder)
43. Page 197, paragraph before the example, second last line: There should be no comma after the x. (Rich Messeder)
44. Page 208: The + or - on the first line is not aligned with text below. (Rich Messeder)
45. Page 208: Under MATRIX 5, double bullets. (Rich Messeder)
46. Page 216: 19 f DIM (i): (i) is on new line, in original manual, all is on one line. (Michael Kussmaul)
47. Page 231: In the code, $e^{(-|x|)}$ is not on the line containing "004- 12". Lines 007 and 011 are similarly misaligned. (Rich Messeder)
48. Page 239, fourth paragraph: "more accurately than is justify by the problem" > "more accurately than is justified by the problem". (Rich Messeder)
49. Page 247, fourth paragraph "Similarly, if a function value is display" > p247: "Similarly, if a function value is displayed". (Rich Messeder)
50. Page 248, second last line: after "strange" should be a full stop instead of a comma. (Rich Messeder)
51. Page 251, last paragraph, third line: There should be a comma after the "since". (Rich Messeder)
52. Page 252, last paragraph: "Functions that could lead to incorrect results can be identified in simple terms by how rapidly it and its low-order derivatives". "Functions" disagrees in number with "it". Already present in the original version. (Rich Messeder)
53. Page 259: Under Low-Power Indication: "A battery symbol(*) shown in the upper-left corner of the display[..]" - Should be lower-left corner. (Didier Lachieze)
54. Page 261, describing the self-tests that don't actually work. The footnote indicates that the ON + Divide keyboard test should display a 12 when finished, not a 15. (Lincoln R.). This error has already been corrected in Edition 2.4 (Sep. 2011).

55. Page 262: No blank line before I-Key description. (Michael Kussmaul)

56. Page 269: MEM section is written twice (repeatedly). (Michael Kussmaul)

57. Index Entry on Page 275 for „Hyberbolic functions“ should start on a new line. (Jurrie Lulofs)

Especially in Complex-Matrix-Section there are some super-scripts (like Tilde) misaligned. (Michael Kussmaul)

Most i in Matrices are overlapping with text (e.g. page 169, example or 171 in the middle of page). (Michael Kussmaul)

Edited: 25 Aug 2012, 2:07 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)
[\[Previous | Next \]](#)

WP 34S: Integration wrapper

Posted by [Miguel Toro](#) on 9 Oct 2011, 10:26 p.m.

This is a wrapper for the integration function of the WP34s for improving accuracy. It integrates by intervals and cumulates the result, comparing a new result with a previous one, until the desired accuracy is reached using the '[approx]?' instruction, so controlling convergence with the display format. FIX 4 and FIX 5 gives good result with relatively fast execution, depending on the function.

- There is now an iteration limit that can be changed in the program
- It is always possible to control time and accuray with the display format

Usage

1. Place a **numeric** label for the function to be integrated in I. Example: 10 [STO] I
2. Enter inferior limit [ENTER] superior limit
3. [XEQ][ENTER]'INW'

Results:

x: integration result
y: previous result
z: superior limit
t: inferior limit

Resources

Flags
none

Registers

I: function label. It has to be numeric
J: iteration limit
r10-r11: limits
r12: cumulative result
r13: previous cumulative result
r14: interval of integration
r15: interval divider

Label
01,02,03,04,05

Instructions

```
001 LBL'INW'
002 1           // Iteration limit
003 5           //
004 STO J       //
005 R[v]
006 0
```

```

007 STO 12          // init. result registers
008 STO 13
009 2
010 STO 15          // init. interval divider, begining with 2
011 [cmp]x]x[<->] Z
012 [cmp]x]STO 10
013 LBL 01          // integration iteration
014 -
015 ABS             // create first interval
016 RCL/ 15
017 STO 14
018 RCL+ 11
019 RCL 11
020 x[<->] Y
021 LBL 02          // integrate interval
022 x>? 10          // check if all intervals calculated
023 GTO 03
024 [integral][>-]I
025 STO+ 12          // cumulative result of intervals
026 [cmp]x]x[<->] Z // prepare next interval
027 ENTER[^]
028 RCL+ 14
029 GTO 02
030 LBL 03
031 DSZ J           // Tests iteration limit
032 GTO 05
033 GTO 04
034 LBL 05
035 RCL 12          // compare new and previous result
036 x[approx]? 13   // accuracy controlled by FIX
037 GTO 04
038 STO 13          // prepare shorter intervals
039 2
040 STO+ 15
041 CLx
042 STO 12
043 [cmp]x]RCL 10
044 GTO 01
045 LBL 04          // shows results
046 [cmp]x]RCL 10
047 [cmp]x]RCL 12
048 RTN

```

Edited: 1 Nov 2011, 9:50 a.m.

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

WP 34S: MIO matrix helper * edited*

Posted by [Miguel Toro](#) on 8 Oct 2011, 8:53 p.m.

This little program works similar to the MIO application from the PPC ROM documentation. It allows entering and reviewing elements of a matrix in a easy and friendly way. the user can enter or review an entire matrix or individual elements. The matrix descriptor is always in Last X.

New:

2011-10-12: both A and B routines cycle through the matrix until the user press [EXIT]. This way, with B, the user does not have to re-enter the matrix descriptor if he wants to review another random element; with A, he can cycle any times he wants through the entire matrix.

Usage:

Review/Enter the elements of an entire matrix

BB.RRCC // Matrix descriptor

[A]

---> Review and/or enter each element then [R/S] for the next one. The program cycles again through the matrix after the last element. Press [EXIT] to finish.

Review/Enter an individual element of a matrix

BB.RRCC // Matrix descriptor

[B]

Row number (i)

[ENTER]

Column number (j)

[R/S]

Review and/or enter Element value in X

[R/S]

Ask for a new element in the same matrix. Press [EXIT] to finish.

Notes:- The matrix descriptor is in **Last x**. You can recall it and do further editing.

Resources

Flags

13: establishes which routine A or B is executing

Registers

I: Index to acces elements through the matrix

Instructions

```
001 LBL'MIO'
002 LBL A           //-- Works on entire matrix elements
003 SF 13
004 M.ALL
```

```

005 STO I
006 LBL 01          // begin of the loop through the matrix
007 RCL I
008 RCL L
009 M.IJ
010 CL[alpha]      // shows the next element indexes
011 [alpha] (
012 [alpha] IP X
013 [alpha] ,
014 [alpha] IP Y
015 [alpha] )
016 RCL[->]I
017 PROMPT
018 STO[->]I        // Store element to keep any change
019 ISG I
020 GTO 01          // End of the loop through the matrix
021 RCL L
022 FS? 13          // Is it B who called LBL A?
023 GTO A           // Cycles through the same matrix
024 LBL B           //-- Works on one chosen matrix element
025 CF 13
026 CL[alpha]
027 [alpha]'R|C'
028 PROMPT
029 x[->]y
030 RCL Z
031 M.REG
032 STO I
033 GTO 01
034 RCL L
035 GTO B           // Cycles through the elements of a matrix

```

Edited: 24 Oct 2011, 3:44 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP35s date functions: DOW, DDate, Date+, Gregorian to Julian conversions

Posted by [David Persinger \(US\)](#) on 16 Sept 2011, 2:21 p.m.

Background: As an engineer who manages projects, I find that I often had a need to determine the number of days between dates, what the date will be X number of days in the future, or which day of the week a certain date falls. These functions are common in the graphing and financial calculators but are missing in most scientifics. The attached program adds these functions to the HP35s.

The program uses routines published by the US Navy Observatory Astronomical Applications Department website.

The routine is valid for dates from 1801-2099.

Usage:
Convert Gregorian to Julian Date:
MM.DDYYYY XEQ I ENTER
Example:
12.252010
XEQ I
ENTER
2,455,556

Convert Julian to Gregorian Date:
XXXXXXX XEQ J Enter
Example:
24555556
XEQ J
ENTER
12.252010

Day of Week:

This will return a numeric DOW to the stack with 0=Saturday, but I didn't want to have to figure the coded date in my head so I added many additional lines that display the day of week in text. You can choose to delete as you desire.
The memory of the 35s is sufficient to leave these in without significant impact.

MM.DDYYYY XEQ M ENTER
Example:
12.252010
XEQ H
ENTER
SAT (for 1 second)
0 (left on stack)

DDAYS: Calculates number of days between 2 dates.

MM.DDYYYY (first date)
MM.DDYYYY (second date)

XEQ L ENTER

Example:
1.012011
9.152011
XEQ L
ENTER
257

DATE+: Computes date x days in future (or -x days in past).

MM.DDYYYY
X
XEQ M
ENTER

Example:

1.012011
257
XEQ M
ENTER
9.152011

Instructions:
* indicates comment, do not try to enter as code.

I001	LBL I	*MM.DDYYYY TO JULIAN
I002	X<>Y	
I003	STO B	
I004	X<>Y	
I005	ENTER	
I006	IP	
I007	STO M	
I008	Rd	
I009	FP	
I010	100	
I011	x	
I012	ENTER	
I013	IP	
I014	STO D	
I015	Rd	
I016	FP	
I017	10000	
I018	x	
I019	STO Y	
I020	FN=K	
I021	SOLVE J	
I022	0.5	
I023	+	
I024	RCL B	
I025	X<>Y	
I026	RTN	
J001	LBL J	*JULIAN TO MM.DDYYYY
J002	IP	
J003	68569	
J004	+	
J005	STO L	
J006	4	
J007	x	
J008	146097	
J009	INT/	
J010	STO N	
J011	RCL L	
J012	X<>Y	
J013	146097	
J014	x	
J015	3	
J016	+	
J017	4	
J018	INT/	
J019	-	
J020	STO L	

```
J021 1
J022 +
J023 4000
J024 x
J025 1461001
J026 INT/
J027 STO I
J028 RCL L
J029 X<>Y
J030 1461
J031 x
J032 4
J033 INT/
J034 -
J035 31
J036 +
J037 STO L
J038 80
J039 x
J040 2447
J041 INT/
J042 STO J
J043 RCL L
J044 x<>y
J045 2447
J046 x
J047 80
J048 INT/
J049 -
J050 STO K
J051 RCL J
J052 11
J053 INT/
J054 STO L
J055 -12
J056 x
J057 RCL J
J058 2
J059 +
J060 +
J061 RCL N
J062 49
J063 -
J064 100
J065 x
J066 RCL I
J067 +
J068 RCL L
J069 +
J070 STO I
J071 1E6
J072 /
J073 RCL K
J074 100
J075 /
J076 +
J077 +
J078 RTN
K001 LBL K      *Julian conversion equation
K002 J=367xY-IDIV(7x(Y+IDIV(M+9,12)),4)+IDIV(275xM,9)+D+1721013.5
K003 RTN
H001 LBL H      *DAY OF WEEK
H002 ENTER
H003 CF10
H004 XEQ I001
H005 IP
H006 5
H007 -
H008 7
H009 RMDR
H010 SF 10
H011 0
H012 x=y?
H013 GT0 H043
H014 CLx
H015 1
H016 x=y?
H017 GT0 H048
H018 CLx
H019 2
H020 x=y?
```

H021	GT0	H053
H022	CLx	
H023	3	
H024	x=y?	
H025	GT0	H058
H026	CLx	
H027	4	
H028	x=y?	
H029	GT0	H063
H030	CLx	
H031	5	
H032	x=y?	
H033	GT0	H068
H034	CLx	
H035	6	
H036	x=y?	
H037	GT0	H073
H038	EQN	BAD DATA
H039	PSE	
H040	CF10	
H041	Rd	
H042	RTN	
H043	SAT	
H044	PSE	
H045	CF10	
H046	Rd	
H047	RTN	
H048	SUN	
H049	PSE	
H050	CF10	
H051	Rd	
H052	RTN	
H053	MON	
H054	PSE	
H055	CF10	
H056	Rd	
H057	RTN	
H058	TUES	
H059	PSE	
H060	CF10	
H061	Rd	
H062	RTN	
H063	WED	
H064	PSE	
H065	CF10	
H066	Rd	
H067	RTN	
H068	THURS	
H069	PSE	
H070	CF10	
H071	Rd	
H072	RTN	
H073	FRI	
H074	PSE	
H075	CF10	
H076	Rd	
H077	RTN	
L001	LBL L	*DAYS BETWEEN DATES
L002	x<>y	
L003	ST0 T	
L004	x<>y	
L005	XEQ I001	
L006	ST0 S	
L007	RCL T	
L008	XEQ I001	
L009	RCL S	
L010	x<>y	
L011	-	
L012	RTN	
M001	LBL M	*DATE+
M002	ST0 S	
M003	x<>y	
M004	XEQ I001	
M005	RCL S	
M006	+	
M007	XEQ J001	
M008	RTN	

Edited: 16 Sept 2011, 9:26 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous \]](#) [\[Next \]](#)

HP-15C LE (Limited Edition) Bug Reports

Posted by [Katie Wasserman](#) on 16 Sept 2011, 11:22 a.m.

This is a working list of bug reports on the newly released HP-15C LE with firmware dated 2011-04-15.

1) PSE run more than once in a program blanks display and run just once causes display flashing issues with SF 9. For example:

```
LBL A
PSE
SF 9
RTN
```

2) When Cx,y/Px,y are run in a program multiple times with certain values (e.g., 8 ,3) it causes display flashing issues.

for example

```
LBL A
STO 0
LBL 9
8
ENTER
3
Cx,x
DSE 0
GTO 9
RTN
```

3) Executing the self tests documented in the manual can corrupt program memory. In addition to the manual, Error 9 is cited on the back plate of the calculator. Perhaps the most expedient solution would be to make the self tests work on the calculator as they do on the 12C+.

4) Current draw is over 20ma when holding down a key, leading to rapid battery depletion. While not a bug this is fixable in firmware and does not happen on the 12C+.

5) The low battery indicator does not appear to work. Further, when the battery voltage gets low enough or the batteries can't supply enough current memory is lost, the calculator will not shut down gracefully.

6) The ON+g+ENTER self test menu has 3 options the first of which turns on all the display segments. If you next press the roll-down key it will turn off half of them. Press it again to turn on the other half. The next press will turn them all back on again. This is all documented.

What's undocumented is what happens if you press and hold the roll-down key. On the 12C+ the display will rapidly cycle through the 3 displays a couple of times then terminate the self test showing the X register. On the 15C LE it will do the same then show the current program line. When you release the roll-down key it will run the program starting at that point.

and from Jeff O. we have:

7) In complex mode, press f - (i) to temporarily display the imaginary component. While the temporary display is up, press a single digit key, as if you were starting to key in a number. As soon as the temporary display of the imaginary component goes away, the digit you pressed will be displayed, and the display will be flashing. Pressing another digit key will continue entry of a multi-digit number beginning with the one that was already entered, and stop the flashing display. Pressing any function key will execute the function on the flashing number and stop the flashing display. If you press more than one digit key while the temporary display is up, the flashing will not occur. On my 15C Classic, key presses during the temporary display are simply ignored.

8) In any mode, press f - PREFIX to temporarily display the full mantissa. While the temporary display is up, press a single digit key, and you will get the same behavior as above.

9) In any mode, press g - MEM to temporarily display the memory allocation. While the temporary display is up, press a single digit key, and you will get the same behavior as above.

10) I tried to think of other temporary display modes which might exhibit the above behavior and could not think of any. You can of course execute PSE from the keyboard which is a temporary mode of sorts. On the 15C Classic, keyboard input during the pause is ignored. On the 15c LE, input will be accepted as in the above examples and displayed after the Pause ends, but the display does not flash.

I point out the above to hopefully aid in producing a new bug-free ROM by documenting all inconsistencies found with the current one.

Edited: 26 Dec 2011, 5:23 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Simple TVM Routines for HP-15C

Posted by [Chris McCormack](#) on 16 Sept 2011, 8:46 a.m.

HP-15C - Simple Time Value of Money

Chris McCormack - 21 Oct 2007

Background

The underlying routine here calculates the present value annuity (PVAf) based on the number of periods and the interest rate per period. Multiplying the payment by PVAf gives the total loan amount, while dividing the loan by the PVAf gives the payment size.

I've had this program sitting in my HP15C for a long time now. I've also used close relatives on the HP11C and HP33S. Not as fancy as some of the TVM programs people use, but nice to have ready when trying to talk turkey at a car dealership!

Description

These routine perform loan calculations using the present value annuity factor, or PVAf. This relates the present value (loan amount) to the periodic payments necessary to pay it off.

$$PVAf(r, N) = (1/r) [1 - 1/(1+r)^N]$$

In this equation, **r** is the decimal interest per payment period (.01 would represent a monthly loan with a 12% rate) and **N** is the number of payments (48 would correspond to a four-year car loan).

Note - Labels 9 and 6 were used because 9 is next to the divide key (breaking the loan down into payments) and 6 is next to the multiply key (building up the total loan amount).

```
001 LBL 9          // ( LoanAmt -- Payment )
002 GSB .9        // divide present value by PVAf
004 RTN

005 LBL 6          // ( Payment -- LoanAmt )
006 GSB .9        // multiply payment by PVAf
008 RTN

009 LBL .9         // ( -- PFAV )
010 RCL 0 1 +      // R0 holds interest per period
013 LN RCL 1 CHS *  // R1 holds number of periods
017 e^x 1 x><y
020 - RCL 0 /
023 RTN
```

Resources Used

Memory : 23 steps

Registers :

R0 : interest rate per period

R1 = number of periods

Labels :

LBL 9 : calculate payment for a given amount borrowed

LBL 6 : calculate amount borrowed for a given payment

LBL.9 : (internal) determine PVAF

Sample calculations :

\$5000 borrowed at 10% with 36 monthly payments --> \$161.34 / month

```
10 ENTER 1200 / STO 0
36 STO 1
5000 GSB 9
```

\$1500/month on a 30 year mortgage at 7.5% --> \$215,526.44 borrowed

```
7.5 ENTER 1200 / STO 0
30 12 * STO 1
1500 GSB 6
```

Note

You can find the necessary interest rate to match payment and loan amounts by using the **SOLVE** function along with six more memory steps, one label, and one register. For example, find out what interest rate would allow a 1500/month payment to handle a 3 year, 5000 loan.

```
023 LBL A // can use any label
024 STO 0 // save current guess
025 GSB .9 RCL 2 - // can use any register except R0 and R1
028 RTN

5000 ENTER 1500 / // find PVAF needed (33.33)
STO 2 36 STO 1 // save PVAF and number of periods
.005 SOLVE A // won't work without good initial guess
1200 * // convert to annual rate (5.06%)
```

Final Note

As noted in the *HP-15C Advanced Functions Handbook*, the math used for computing the annuity factors can suffer from roundoff errors when very low interest rates apply over a large number of periods. Using the technique outlined on page 181 of the handbook, an alternative subroutine .9 finds the correct answer.

```
009 LBL .9 // ( -- PVAF )
010 STO I RCL 0 ENTER ENTER // save incoming x-register
014 1 + LN X><Y
018 LASTx 1 TEST 6 - // TEST 6 (X<>Y)
022 / * RCL 1 *
026 e^x 1/x 1 X><Y
030 - RCL 0 /
033 RCL I X><Y // restore X value to stack (above PVAF)
035 RTN
```

Running the example from the handbook, with 11.25% annual interest for payments each second (31,536,000 periods!) generates a present value annuity factor of 29,826,791.69 (instead of 33,221,310.84 returned by the original LBL .9 routine).

\\ 20110916 - cjmcc

Edited: 16 Sept 2011, 9:01 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Creating ROM image files of an HP-41 module

Posted by [Juergen Keller](#) on 9 Aug 2011, 4:55 p.m.

This is the procedure I use to dump a HP-41 module to .rom files (which then can be used, e.g, to burn a Clonix). Note that there are many other ways to do it ...

1. Requirements

- HP-41 with capability to save a 4k page to a mass storage device, e.g.:
 - HP-41CL with CCD/OSX (SAVEROM)
 - HP-41CY (WRTPG)
 - HP-41 with CCD OS/X provided by a Clonix module
 - HP-41 with HEPAX (WRTROM) provided by a NoVRAM etc.
 - etc.
- HP 82160A HP-IL interface
- Windows PC (drivers required by PIL-Box need to be installed)
- [PIL-Box](#) device
- [ILPer](#) program
- [HPDir](#) program
- [lif2rom](#) program

2. Preparations

1. Insert the module to dump (and any other modules needed by your setup, e.g. Clonix) into the HP-41.
2. Connect the PIL-Box to the PC, and the HP-41 to the PIL-Box.

3. Dump all pages of the module to a LIF file

1. Run ILPer (select mass storage LIF file, appropriate port and baud rate) and start the PIL-Box interface.
2. Turn on the HP-41.
3. Enter a file name for the page into the ALPHA register.
4. Enter the page number (8-15) into the X register.
5. Execute the SAVEROM function (or WRTPG, WRTROM, ...).
6. Repeat steps 3 to 5 for all pages of the module.
7. Stop the PIL-Box and quit ILPer.

4. Extract all pages of the module from the LIF file and convert them to ROM files

1. Open a command prompt.
2. Run Hpdirextract <file> <page>, where <file> is the name of the LIF file (as specified in step 3.1) and <page> is the name of the page (as entered in step 3.3).
3. Run lif2rom <page>.#e070 <name>.rom.
4. Repeat steps 2 to 3 for all pages of the module.

Any comments are welcome.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

Thank You HP Users

Posted by [LouisD](#) on 9 Aug 2011, 3:21 p.m.

I have a 20S that I've used for years until just recently when it began to miss key entries. I noticed that if I twisted the case slightly that it functioned perfectly. With the help of an article here, I drilled out the stakes and opened the case. Nothing stood out that was "loose" so I hit upon a simple solution, I folded a piece a paper and place it inside the case to cause the unit to twist slightly when reassembled. PRESTO - a fully functional HP 20S

Rube Goldberg fix, but I think it's good for another decade!

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

EE calculations, some not obvious (reformatted)

Posted by [Thomas Klemm](#) on 26 May 2011, 5:14 p.m.

This is a reformatted copy of the article [EE calculations, some not obvious](#).

HEWLETT PACKARD CALCULATIONS

I used HP RPN pocket calculators for electronic solutions, and I wanted to find a fast way to solve common problems using as few keystrokes as possible, and getting maximum information during the process.

This method uses the repeating t register.

Formulas now in use were derived for pen and pencil mathematics, totally outmoded by the calculator.

This is the keystroke procedure for quick solutions.

Resistive Dividers

When calculating from known resistors, enter the lower resistor repeatedly.

- Key in the top resistor, and add, the result is the total resistance,
- then divide, the result is the fractional output voltage of the divider.
- Then multiply and subtract to see the output resistance from the divider tap.

To design such a network, key in the output resistance desired, and enter repeatedly.

- Key in the division fraction, and divide to see the upper resistance.
- Divide again to restore the fraction, key 1, subtract, change sign, divide, and see the lower resistor.
- The relation between the upper resistor and the output resistance is the fraction of the divider.
- 1 minus the fraction is the relation between the output resistance and the lower resistance.
- Given the fraction and either resistor, the other value may be found.

I have used this method to find the added end resistors for setting particular potentiometer control ranges.

The difference between the two potentiometer fractions desired at the potentiometer ends, relates the potentiometer resistance to the total resistance.

Thus a particular value of potentiometer may be used to obtain the desired range of fractions.

EXAMPLE

It was desired to use a 1K potentiometer to adjust a power supply to vary between 24 and 16 volts.

The IC regulated at 4.75 volts from the potentiometer.

It is desired to calculate the end resistors needed to set the correct range.

- $4.75/24 = 0.2$ store 1.

- $4.75/16 = 0.3$ store 2.
- Recall 1, subtract, see 0.1,
- key 1000 swap X&Y,
- divide, see 10105, the total resistance.
- Enter several times.
- Recall 1, multiply, see 2000, the resistance when the pot is at the lower extreme,
- add 2000 ohms to the lower end of the pot.
- Cl X,
- recall 2,
- multiply, see 3000, this is the sum of the pot resistance and the pot lower end resistor.

The upper end resistor must be 10205 minus 3000 = 7105. Some compromise values may be used.

Transistor bias

I was solving for a transistor bias divider, and wondered how to account for the base current drop in the divider. I thought to divide the divider output resistance by Beta and add it to the actual emitter resistor. To do this procedure, solve the bias divider resistors, storing the divider fraction.

- With R0 in the register key Beta, divide, and see the emitter resistor equivalent to be added.
- Key the actual emitter resistor and add, see the total emitter resistor, accounting for the bias source drop.
- Key the source voltage;
- recall the fraction, multiply and see the divider voltage.
- Key 0.7, subtract, see the emitter voltage.
- Interchange X&Y and divide, to see the emitter current:
- check by multiplying by the collector resistor, to see the collector resistor drop.

Summing network

The resistive divider can be extended to a more general case.

I needed to design a telemetry output that combined several signals, to a common output, having a 10 K output resistance.

The signal of interest was a +/- 12 volt VCO control voltage.

The telemetry output was to be 0-5 volts, using the range 0.5 to 4.5 volts normally.

The centering voltage was 2.5 volts from a regulated 12 volts supply.

A third resistor to ground gave the desired output resistance.

I found a general solution.

Each resistor is determined by the fraction of its input voltage contributed to the output sum, thus the telemetry signal was to output 1/6th of its signal.

The resistor was then 10K times 6 or 60K.

The centering voltage was 2.5 volts derived from +12, a ratio of 4.8 times, 48K.

The third resistor was found by subtracting the other two fractions from 1: $1 - 1/6 - 1/4.8$, a fraction of 0.625 and dividing the output resistance to get a resistance of 16K.

Some compromise values were used, which proved satisfactory.

RC frequency corners

The process may be simplified as $2 \pi F R C = 1$.

The known values are multiplied, and the product inverted to give the missing element.

For constant frequency solutions enter $2 \pi F$ repeatedly.

Tuned circuits

Here intermediate results in the solution give other useful information.

- Key in L , then C , and store C in a register,
- divide and take the square root; this is the impedance of the resonant elements.
- Recall the capacitance value and multiply,
- then multiply by 2π and invert to see the frequency in Hertz.

This is the frequency corner formula using R_0 as the R value.

Q if known may be used to compute the resonant impedance, or the Q may be found by dividing the parallel circuit damping resistance by the impedance, or the impedance by the series circuit resistance.

The square root of L/C is also the output resistance of a simple LC filter, such as a power supply, and can be used to estimate transient voltages from current changes.

Resonant values

When using a constant frequency, resonant values may be calculated quickly.

- Key in $2 \pi F$ and enter it repeatedly.
- Then key in either L or C ,
- multiply twice, and invert to see the other resonant circuit element.
- When $2 \pi F L$ is shown, it is the reactance of the circuit values.

This procedure gives quick trials of various circuit values, give impedance values for filter and by-pass elements, and value choices for a given Q .

Ohms law

This may seem simple, but can be used as a rapid verification of dissipation and the voltage and current relation. Remember the little circles of voltage E divided by the $I R$ product.

- Key in the voltage repeatedly, and divide by either I or R to see the other value, as in $E = I R$.
- The power in a resistor can be found by keying in the resistor and dividing to get I , then multiplying to get $E I$.
- It can be used to solve $W = E I$, by keying in power repeatedly, and dividing to get either E or I .

These methods allow such rapid solutions; they will increase your ability to investigate possibilities in your designs. Practice will allow fast and sure answers to a wide range of problems. Familiarity will allow you to extend these methods to your particular problems.

I trained one tech to calculate time payments faster than they could be found in a book.

E. Samuel Levy
754 Temple St.
San Diego 92106
619-223-6292

designnut@cox.net

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Multiple Period Percentage Change Program for the 30b

Posted by [Mark Harman](#) on 25 May 2011, 12:13 a.m.

Hello everyone,

Previously, I posted a program that add a top-level shifted percentage change function to the 30b while using the RPN stack.

I decided that this was not enough. If amounts change over multiple periods, discounting or compounding take place. The best way to solve this dilemma is to incorporate this into a brand new program.

Here it goes:

SH% (put this on line zero to assign the program to this key combination)

STO 1

Roll down

swap

/

RCL 1

1/x

y^X

1

-

100

*

(you can add a message like delta (the symbol) % or Pct chng here if you want)

The way this program works is such:

Say you had \$50 two years ago and you have \$90 today. You want to know what the percentage increase is over these two years:

50 input 90 input 2 SH% (SH is shift hold)

You should get an answer of 34.16

Now say you have the opposite situation and want to know the percentage decrease.

90 input 50 input 2 SH%

You should get an answer of -25.46440

Keep in mind that if you have only one period, you MUST enter a 1 for the third argument - otherwise you will get an error. This program requires three arguments to work.

(Now, if your logic works such that you prefer to have the new amount first, then get rid of the swap command and then enter the new first, old second, and # periods third.)

Hope this helps anyone who doesn't like going through the %Calc menu to get to this function.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 30b programs for interest rate conversions with negative or fractional periods per year

Posted by [gene wright](#) on 3 May 2011, 2:30 p.m.

These programs allow the use of negative values for "Periods per year" when converting interest rates. They also allow the use of fractional periods per year. End of period compounding assumed.

First program: Computing the effective rate. Inputs: Nominal rate, =, number of periods.

```
/
Ans
) Swap
1
+
) Swap
Y^X
1
-
Shift+Stop
```

When run from the program catalog, key 0.1224 = 2 +/- = =

Result shown is 0.1346

2nd program: Effective rate to nominal rate. Inputs: Effective rate, =, number of periods.

```
=
=
shift+UP
1
+
) Swap
1/X
Y^X
1
-
*
Shift+STOP
```

Inputs: 0.1346 = 2 +/- = =

Outputs 0.1224

Inputs: 0.1346 = 360 = 210 / =

Outputs: 0.1310

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

Game for HP-35s

Posted by [Paul Dale](#) on 16 Mar 2011, 5:46 a.m.

A dragon has kidnapped the beautiful princess and you are tasked with her rescue. You will have to brave all manner of nasty beasts on your quest.

The quest is a series of meet monster, kill or run away from monster cycles. When you kill a monster you will usually gain some treasure and possibly a magic item. You might also get healed. The difficulty of the monsters generally increases with your abilities. Once you defeat the dragon, you win the game. If you've accumulated sufficient treasure, you'll marry the princess as well.

When displaying numbers they are viewed via registers:

E = experience (your total and current earnings)

H = hit points (both damage and your total)

G = gold

L = level

S = spells

Periodically you'll be asked one of two questions:

"attack or flee" (0=ATK 1=FLEE)

"sword or spell" (0=SWD 1=SPELL)

In each case enter a zero number and press R/S for the first and enter a non-zero number and press R/S for the second. By default a zero will be entered for you so you only actually need to press R/S in that case.

D001	LBL	D	D238	GTO	D381	D475	XEQ	D065
D002	ALL		D239	XEQ	D075	D476	XEQ	D029
D003	XEQ	D702	D240	x<y?		D477	RCL	M
D004	SF	10	D241	GTO	D393	D478	XEQ	D067
D005	XEQ	D639	D242	x>y?		D479	/	
D006	STO	M	D243	GTO	D345	D480	+	
D007	XEQ	D639	D244	eqn	TITAN	D481	2.5	
D008	STO	N	D245	PSE		D482	-	
D009	XEQ	D639	D246	[16,10,50]		D483	IP	
D010	STO	0	D247	[2,16,20]		D484	FS?	1
D011*	XEQ	D063	D248	[1,22,25]		D485	XEQ	D509
D012	RCL	H	D249	GTO	D399	D486	x<=0?	
D013	x>=y?		D250*	XEQ	D078	D487	XEQ	D071
D014	GTO	D110	D251	x<y?		D488	GTO	D533
D015	XEQ	D057	D252	GTO	D333	D489*	eqn	MISSED
D016	x<=y?		D253	x>y?		D490	PSE	
D017	GTO	D081	D254	GTO	D387	D491	GTO	D535
D018	RCL	K	D255	eqn	ENT	D492*	XEQ	D069
D019	XEQ	D057	D256	PSE		D493	/	
D020	XEQ	D069	D257	[9,10,40]		D494	XEQ	D073
D021	RCL	H	D258	[2,6,10]		D495	+	
D022	y^x		D259	[2,15,25]		D496	IP	
D023	*		D260	GTO	D399	D497	x<=0?	
D024	XEQ	D063	D261*	eqn	GIANT BAT	D498	GTO	D071
D025	-		D262	PSE		D499	RTN	
D026	x>=y?		D263	[1,2,0]		D500*	0.9	
D027	GTO	D110	D264	[1,4,0]		D501	*	
D028	GTO	D088	D265	[1,0,12]		D502	RTN	

D029*	RANDOM	D266	GT0 D399	D503*	XEQ D067
D030	*	D267*	eqn GOBLIN	D504	-
D031	INTG	D268	PSE	D505	RTN
D032	XEQ D071	D269	[1,8,0]	D506*	XEQ D067
D033	+	D270	[1,8,0]	D507	+
D034	RTN	D271	[1,2,15]	D508	RTN
D035*	ST0 A	D272	GT0 D399	D509*	XEQ D057
D036	CF 10	D273*	eqn SKELETON	D510	XEQ D029
D037*	eqn REGY*RAND	D274	PSE	D511	+
D038	INTG	D275	[1,8,2]	D512	XEQ D065
D039	+	D276	[1,8,0]	D513	+
D040	DSE A	D277	[2,1,18]	D514	FS? 4
D041	GT0 D037	D278	GT0 D399	D515	GT0 D517
D042	SF 10	D279*	eqn GIANT RAT	D516	RTN
D043	X<>Y	D280	PSE	D517*	XEQ D063
D044	RV	D281	[1,6,0]	D518	XEQ D069
D045	RTN	D282	[1,6,0]	D519	XEQ D029
D046*	XEQ D633	D283	[1,1,13]	D520	+
D047	XEQ D035	D284	GT0 D399	D521	XEQ D057
D048	+	D285*	eqn DWARF	D522	+
D049	RTN	D286	PSE	D523	RTN
D050*	XEQ D069	D287	[2,8,0]	D524*	XEQ D071
D051	XEQ D029	D288	[1,10,1]	D525	ST0- L
D052	XEQ D071	D289	[1,2,20]	D526	eqn ZOT!
D053	-	D290	GT0 D399	D527	PSE
D054	RTN	D291*	eqn GIANT SPIDER	D528	XEQ D067
D055*	XEQ D063	D292	PSE	D529	XEQ D069
D056	GT0 D029	D293	[2,8,4]	D530	+
D057*	10	D294	[1,10,8]	D531	RCL H
D058	RTN	D295	[1,4,17]	D532	XEQ D035
D059*	3	D296	GT0 D399	D533*	ST0- D
D060	RTN	D297*	eqn ZOMBIE	D534	XEQ D613
D061*	100	D298	PSE	D535*	RCL D
D062	RTN	D299	[3,8,0]	D536	x<=0?
D063*	20	D300	[1,6,2]	D537	GT0 D568
D064	RTN	D301	[2,5,8]	D538	RCL B
D065*	8	D302	GT0 D399	D539	FS? 3
D066	RTN	D303*	eqn GHOST	D540	XEQ D492
D067*	4	D304	PSE	D541	ST0 Q
D068	RTN	D305	[10,8,50]	D542*	XEQ D055
D069*	2	D306	[2,4,-1]	D543	RCL+ F
D070	RTN	D307	[2,16,30]	D544	RCL N
D071*	1	D308	GT0 D399	D545	XEQ D067
D072	RTN	D309*	eqn DAEMON	D546	/
D073*	0.5	D310	PSE	D547	-
D074	RTN	D311	[8,8,20]	D548	XEQ D057
D075*	XEQ D069	D312	[1,8,2]	D549	-
D076	+	D313	[2,11,26]	D550	FS? 2
D077	RTN	D314	GT0 D399	D551	XEQ D503
D078*	XEQ D069	D315*	eqn GNOME	D552	x<0?
D079	-	D316	PSE	D553	GT0 D565
D080	RTN	D317	[3,10,0]	D554	RCL C
D081*	RCL K	D318	[1,8,1]	D555	XEQ D046
D082	-9	D319	[1,7,18]	D556	FS? 0
D083	RCL+ H	D320	GT0 D399	D557	XEQ D500
D084	10220	D321*	eqn BASILISK	D558	INTG
D085	*	D322	PSE	D559	x<=0?
D086	x>=y?	D323	[6,8,4]	D560	XEQ D071
D087	GT0 D110	D324	[2,10,10]	D561	ST0- J
D088*	XEQ D067	D325	[1,13,24]	D562	eqn OUCH!
D089	XEQ D029	D326	GT0 D399	D563	PSE
D090	XEQ D065	D327*	eqn SLIME	D564	XEQ D613
D091	+	D328	PSE	D565*	DSE Q
D092	RCL 0	D329	[5,10,20]	D566	GT0 D542
D093	XEQ D059	D330	[1,4,0]	D567	GT0 D425
D094	/	D331	[4,8,12]	D568*	eqn KILLED!
D095	+	D332	GT0 D399	D569	PSE
D096	IP	D333*	eqn DEVIL	D570	XEQ D067
D097	x<=0	D334	PSE	D571	XEQ D069
D098	XEQ D071	D335	[10,8,30]	D572	RCL E
D099	ST0+ I	D336	[1,10,5]	D573	y^x
D100	ST0+ J	D337	[2,14,24]	D574	RCL* P
D101	XEQ D071	D338	GT0 D399	D575	RCL* B
D102	ST0+ H	D339*	eqn BARBARIAN	D576	ST0+ K
D103	ST0+ L	D340	PSE	D577	XEQ D608
D104	XEQ D050	D341	[4,12,16]	D578	XEQ D059
D105	ST0+ M	D342	[1,10,2]	D579	RCL E
D106	XEQ D050	D343	[1,7,13]	D580	y^x
D107	ST0+ N	D344	GT0 D399	D581	XEQ D029
D108	XEQ D050	D345*	eqn VAMPIRE	D582	ST0+ R
D109	ST0+ 0	D346	PSE	D583	XEQ D057

D110*	RCL	H	D347	[8,10,10]	D584	/
D111	XEQ	D623	D348	[2,12,8]	D585	IP
D112	RCL	K	D349	[1,15,24]	D586	STO+ K
D113	XEQ	D608	D350	GT0 D399	D587	XEQ D618
D114	RCL	J	D351*	eqn 00ZE	D588	RANDOM
D115	XEQ	D613	D352	PSE	D589	0.1
D116	RCL	L	D353	[12,10,30]	D590	RCL* E
D117	XEQ	D628	D354	[1,6,0]	D591	x>y?
D118	FS?	4	D355	[5,9,14]	D592	XEQ D645
D119	GT0	D693	D356	GT0 D399	D593*	XEQ D057
D120	XEQ	D065	D357*	eqn MOLD MONSTER	D594	1/x
D121	XEQ	D067	D358	PSE	D595	RANDOM
D122	*		D359	[2,8,0]	D596	x>=y?
D123	XEQ	D059	D360	[1,2,0]	D597	GT0 D011
D124	XEQ	D071	D361	[3,3,10]	D598	eqn HEAL
D125	RCL+	H	D362	GT0 D399	D599	PSE
D126	*		D363*	eqn OGRE	D600	RCL I
D127	x>y?		D364	PSE	D601	STO J
D128	x<>y		D365	[5,12,10]	D602	RANDOM
D129	XEQ	D029	D366	[2,8,8]	D603	0.3
D130	STO	E	D367	[1,10,21]	D604	+
D131	XEQ	D067	D368	GT0 D399	D605	IP
D132	x^2		D369*	eqn GIANT SNAKE	D606	STO+ L
D133	x<y?		D370	PSE	D607	GT0 D011
D134	GT0	D199	D371	[4,8,8]	D608*	x<> E
D135	x=y?		D372	[2,8,4]	D609	VIEW E
D136	GT0	D369	D373	[1,5,15]	D610	PSE
D137	XEQ	D065	D374	GT0 D399	D611	x<> E
D138	-		D375*	eqn TROLL	D612	RTN
D139	x<y?		D376	PSE	D613*	x<> H
D140	GT0	D171	D377	[6,10,40]	D614	VIEW H
D141	x=y?		D378	[1,6,6]	D615	PSE
D142	GT0	D357	D379	[2,9,22]	D616	x<> H
D143	XEQ	D067	D380	GT0 D399	D617	RTN
D144	-		D381*	eqn ELEMENTAL	D618*	x<> G
D145	x<y?		D382	PSE	D619	VIEW G
D146	GT0	D160	D383	[10,12,30]	D620	PSE
D147	x=y?		D384	[1,12,10]	D621	x<> G
D148	GT0	D267	D385	[2,16,17]	D622	RTN
D149	XEQ	D078	D386	GT0 D399	D623*	x<> L
D150	x<y?		D387*	eqn WYVERN	D624	VIEW L
D151	GT0	D279	D388	PSE	D625	PSE
D152	x>y?		D389	[7,12,16]	D626	x<> L
D153	GT0	D261	D390	[1,8,8]	D627	RTN
D154	eqn	KOBOLD	D391	[2,14,28]	D628*	x<> S
D155	PSE		D392	GT0 D399	D629	VIEW S
D156	[1,4,1]		D393*	eqn DRAGON	D630	PSE
D157	[1,6,-1]		D394	PSE	D631	x<> S
D158	[1,0,13]		D395	[24,20,100]	D632	RTN
D159	GT0	D399	D396	[1,20,30]	D633*	CF 10
D160*	XEQ	D075	D397	[2,30,29]	D634	eqn [0,0,1]*REGX
D161	x<y?		D398	SF 4	D635	eqn [0,1,0]*REGY
D162	GT0	D285	D399*	STO B	D636	eqn [1,0,0]*REGZ
D163	x>y?		D400	Rv	D637	SF 10
D164	GT0	D273	D401	STO C	D638	RTN
D165	eqn	ORC	D402	Rv	D639*	XEQ D063
D166	PSE		D403	XEQ D046	D640	RANDOM
D167	[1,12,4]		D404	STO D	D641	SQRT
D168	[1,8,3]		D405	STO P	D642	*
D169	[1,3,17]		D406	RCL E	D643	IP
D170	GT0	D399	D407	XEQ D059	D644	RTN
D171*	XEQ	D067	D408	/	D645*	XEQ D069
D172	+		D409	+/-	D646	XEQ D067
D173	x>y?		D410	INTG	D647	XEQ D029
D174	GT0	D188	D411	ABS	D648	XEQ D071
D175	x=y?		D412	STO E	D649	-
D176	GT0	D315	D413	RCL B	D650	x=0?
D177	XEQ	D075	D414	XEQ D633	D651	GT0 D666
D178	x<y?		D415	STO B	D652	x<y?
D179	GT0	D339	D416	Rv	D653	GT0 D673
D180	x>y?		D417	STO F	D654	x=y?
D181	GT0	D327	D418	Rv	D655	GT0 D680
D182	eqn	GNOLL	D419	STO G	D656	FS? 3
D183	PSE		D420	GT0 D428	D657	RTN
D184	[4,10,4]		D421*	CF 4	D658	SF 3
D185	[1,10,1]		D422*	eqn GOT AWAY	D659	eqn MAGIC HELMET
D186	[1,6,17]		D423	PSE	D660	PSE
D187	GT0	D399	D424	GT0 D593	D661	XEQ D071
D188*	XEQ	D078	D425*	RCL J	D662*	1e3
D189	x<y?		D426	x<=0?	D663	*
D190	GT0	D297	D427	GT0 D690	D664	STO+ R

D191	x>y?	D428*	eqn 0=ATK 1=FLEE	D665	RTN
D192	GT0 D291	D429	PSE	D666*	FS? 0
D193	eqn GIANT LEECH	D430	CLSTK	D667	RTN
D194	PSE	D431	STOP	D668	SF 0
D195	[2,8,4]	D432	x=0?	D669	eqn MAGIC SHIELD
D196	[1,8,4]	D433	GT0 D449	D670	PSE
D197	[1,6,11]	D434	FS? 4	D671	XEQ D069
D198	GT0 D399	D435	GT0 D421	D672	GT0 D662
D199*	XEQ D065	D436	XEQ D057	D673*	FS? 1
D200	+	D437	RCL* E	D674	RTN
D201	x<y?	D438	XEQ D069	D675	SF 1
D202	GT0 D233	D439	RCL* N	D676	eqn MAGIC SWORD
D203	x=y?	D440	-	D677	PSE
D204	GT0 D321	D441	RCL+ H	D678	XEQ D057
D205	XEQ D067	D442	XEQ D061	D679	GT0 D662
D206	-	D443	/	D680*	FS? 2
D207	x<y?	D444	RANDOM	D681	RTN
D208	GT0 D222	D445	x>=y?	D682	SF 2
D209	x=y?	D446	GT0 D422	D683	eqn MAGIC ARMOUR
D210	GT0 D375	D447	eqn CAUGHT YOU!	D684	PSE
D211	XEQ D078	D448	PSE	D685	XEQ D067
D212	x<y?	D449*	RCL L	D686	GT0 D662
D213	GT0 D363	D450	x=0?	D687*	eqn MARRY PRINCESS
D214	x>y?	D451	GT0 D458	D688	PSE
D215	GT0 D351	D452	eqn 0=SWD 1=SPELL	D689	RTN
D216	eqn BUGBEAR	D453	PSE	D690*	eqn YOU DIED!
D217	PSE	D454	CLSTK	D691	PSE
D218	[6,8,15]	D455	STOP	D692	GT0 D702
D219	[1,6,4]	D456	x<>0?	D693*	eqn BEAT DRAGON
D220	[4,9,20]	D457	GT0 D524	D694	PSE
D221	GT0 D399	D458*	XEQ D055	D695	XEQ D061
D222*	XEQ D075	D459	RCL H	D696	x^2
D223	x<y?	D460	XEQ D069	D697	RCL R
D224	GT0 D309	D461	/	D698	x>=y?
D225	x>y?	D462	+	D699	XEQ D687
D226	GT0 D303	D463	RCL M	D700	eqn YOU WIN!
D227	eqn GIANT	D464	XEQ D067	D701	PSE
D228	PSE	D465	/	D702*	CLVARS
D229	[6,12,30]	D466	+	D703	CLSTK
D230	[2,10,12]	D467	IP	D704	CF 0
D231	[1,12,23]	D468	FS? 1	D705	CF 1
D232	GT0 D399	D469	XEQ D506	D706	CF 2
D233*	XEQ D067	D470	RCL G	D707	CF 3
D234	+	D471	x>y?	D708	CF 4
D235	x>y?	D472	GT0 D489	D709	CF 10
D236	GT0 D250	D473	eqn HIT	D710	RTN
D237	x=y?	D474	PSE		

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

N-queens benchmark and the HP-17BII

Posted by [Thomas Klemm](#) on 7 Feb 2011, 5:48 a.m.

Equation for the 8-queens problem

```

QUEENS:
Q=A+B+C+D+E+F+G+H+
Σ(I:1:8:1:
Σ(J:1:8:1:
IF(I<>J AND SQ(I-J)<>1:
Σ(K:1:8:1:
IF(I<>K AND SQ(I-K)<>4
AND J<>K AND SQ(J-K)<>1:
Σ(L:1:8:1:
IF(I<>L AND SQ(I-L)<>9
AND J<>L AND SQ(J-L)<>4
AND K<>L AND SQ(K-L)<>1:
Σ(M:1:8:1:
IF(I<>M AND SQ(I-M)<>16
AND J<>M AND SQ(J-M)<>9
AND K<>M AND SQ(K-M)<>4
AND L<>M AND SQ(L-M)<>1:
Σ(N:1:8:1:
IF(I<>N AND SQ(I-N)<>25
AND J<>N AND SQ(J-N)<>16
AND K<>N AND SQ(K-N)<>9
AND L<>N AND SQ(L-N)<>4
AND M<>N AND SQ(M-N)<>1:
Σ(O:1:8:1:
IF(I<>O AND SQ(I-O)<>36
AND J<>O AND SQ(J-O)<>25
AND K<>O AND SQ(K-O)<>16
AND L<>O AND SQ(L-O)<>9
AND M<>O AND SQ(M-O)<>4
AND N<>O AND SQ(N-O)<>1:
Σ(P:1:8:1:
IF(I<>P AND SQ(I-P)<>49
AND J<>P AND SQ(J-P)<>36
AND K<>P AND SQ(K-P)<>25
AND L<>P AND SQ(L-P)<>16
AND M<>P AND SQ(M-P)<>9
AND N<>P AND SQ(N-P)<>4
AND O<>P AND SQ(O-P)<>1:
L(A:I)XL(B:J)XL(C:K)XL(D:L)XL(E:M)XL(F:N)XL(G:O)XL(H:P)/0
:0)):0)):0)):0)):0)):0)):0)))

```

Solution

Solve the equation for Q. After a while (3:29) you get an error message:

SOLUTION NOT FOUND

Now you can recall the solution in the variables A-H. The result { 1, 5, 8, 6, 3, 7, 2, 4 } corresponds to the following

configuration:

	A	B	C	D	E	F	G	H
8			#					
7						#		
6				#				
5		#						
4								#
3					#			
2							#	
1	#							

References

[Eight queens puzzle](#)

[Calculator Speed Benchmark using the N-Queens Problem](#)

Edited: 7 Feb 2011, 6:27 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Sum of Digits Divisibility Challenge - an unusual solution

Posted by [David Hayden](#) on 16 Jan 2011, 9:39 p.m.

On January 5, 2011, Don Shepherd posed the following challenge in the forum. You should be able to find the thread in the archives.

Quote:

A new year deserves a new challenge!

Write a program on whatever calculator you like to answer the following question: Are there 4 consecutive 4-digit integers that are each evenly divisible by the sum of their digits? If so, list them (there MAY be more than one set, and I probably wouldn't pose this challenge if there were no sets).

For example, if 1001 is evenly divisible by 2 and 1002 is evenly divisible by 3 and 1003 is evenly divisible by 4 and 1004 is evenly divisible by 5, you've got a set. Of course, only one of those 4 conditions is true in this example, so 1001-1004 don't work.

So post your program and tell how long it took. This challenge has no practical value that I can think of other than it's just fun to develop the algorithm and get it to work, and that's reason enough for me.

Many clever solutions were posted, but they mostly used and optimized the same basic algorithm:

```
for each number between 1000 and 9999
  if it's divisible by the number of digits then
    if the next 3 are also divisible by the number of digits then
      you have found a solution
```

For fun, I decided to try a completely different approach. I posted my solution on January 16, 2011. Don asked me to post it as an article, so here it is. I want to thank Don for providing this challenge.

Here is a solution to the challenge in User RPL that takes a completely different approach. On a 50g, it runs in 123 seconds. I haven't attempted any coding optimization - the time is due entirely to the algorithm - so I'm sure it can be improved.

Instead of checking each possible 4-digit number to see whether it's divisible by the sum of its digits, this program starts with a possible sum of digits for four consecutive numbers. It then checks the much smaller set of numbers that can be divided by this set.

For example, the sum of the digits in the first solution (beginning at 1014) are 6,7,8, and 9. It turns out that any set of numbers N , $N+1$, $N+2$, $N+3$ that are divisible by 6,7,8 and 9, must be of the form $N = 6 + 504k$ where k is an integer. So for the sums {6 7 8 9} we only have to check every 504'th number.

The program is quite long and this will be a long explanation. I hope you'll find it interesting. I implemented the

program as a directory for simplicity (and debugging!) and I'll refer to the various programs in the directory.

Throughout this explanation, I'll use N , $N+1$, $N+2$ and $N+3$ (or N_0 , N_1 , N_2 and N_3) to refer to the sequential 4-digit numbers, and I'll use s_0 , s_1 , s_2 and s_3 to refer to the sum of the digits in each of these numbers.

Enumerating the sums of digits

To begin, let's figure the possible values for the sum of the digits of four sequential numbers between 1000 and 9999. Clearly the smallest sum is 1 (for 1,000) and the largest sum is 36 (for 9999). So any sum must be between 1 and 36.

For most sequences of N , the sum of their digits is also a sequence. E.g., for $N = \{1001, 1002, 1003, 1004\}$, $s = \{2, 3, 4, 5\}$. So one possible set of sums is:

$$\{s, s+1, s+2, s+3\} \text{ for } s = 1 \text{ through } 33$$

Now consider $N = \{1207, 1208, 1209, 1210\}$. The digit sums are $s = \{10, 11, 12, 4\}$. In general, when the one's digit rolls over from 9 to 0, you lose 9 in the digit sum, but you gain 1 because the 10's digit increments. So another possible set of sums is:

$$\{s, s+1, s+2, s-6\} \text{ for } s = 8 \text{ to } 33$$

The roll-over from 9 to 0 could occur anywhere in the sequence, so other possibilities are:

$$\begin{aligned} &\{s, s+1, s-7, s-6\} \text{ for } s = 9 \text{ to } 34 \\ &\{s, s-8, s-7, s-6\} \text{ for } s = 10 \text{ to } 35 \end{aligned}$$

The numbers in N can also roll over from 99 to 00 or 999 to 000, these give the possible sums:

$$\begin{aligned} &\{s, s-17, s-16, s-15\} \text{ for } s = 19 \text{ to } 35 \\ &\{s, s+1, s-16, s-15\} \text{ for } s = 18 \text{ to } 34 \\ &\{s, s+1, s+2, s-15\} \text{ for } s = 17 \text{ to } 33 \end{aligned}$$

$$\begin{aligned} &\{s, s-26, s-25, s-24\} \text{ for } s = 27 \text{ to } 36 \\ &\{s, s+1, s-25, s-24\} \text{ for } s = 26 \text{ to } 35 \\ &\{s, s+1, s+2, s-24\} \text{ for } s = 25 \text{ to } 34 \end{aligned}$$

In my solution, all of these lists are calculated by the SUMS program. The result of SUMS is a list of lists where each sublist is 4 numbers that represent the sums of the digits of 4 consecutive numbers.

Restricting the start and end values

The challenge asks us to find a 4-digit number - one that's between 1000 and 9999, but if we start with a possible sum of the digits in N , we can use that information to restrict the upper and lower bounds of the search. For example, suppose we're considering the sums 10, 11, 12, and 13. The smallest 4-digit number whose digits add up to 10 is 1009 and the largest is 9001, so there's no point in checking numbers less than 1009 or larger than 9001.

The subprogram BIGandSMALL calculates the largest and smallest 4-digit numbers whose digits sum to each of the values between 1 and 36. It leaves the results on the stack in two arrays. The main program calls BIGandSMALL once and stores the results in global variables BIGGEST and SMALLEST. For example, BIGGEST[12] is the biggest 4 digit number whose digits sum to 12. Later when we check a particular set of sums, we use BIGGEST and SMALLEST to get the upper and lower bounds of the loop.

Mathematics

Suppose we're given a list $\{s_0, s_1, s_2, s_3\}$ that is a solution for the sequence of numbers $\{N, N+1, N+2, N+3\}$. That means that N is evenly divisible by s_0 , $N+1$ is evenly divisible by s_1 etc. Mathmatically:

$$\begin{aligned} N &== 0 \bmod s_0 \\ N+1 &== 0 \bmod s_1 \\ N+2 &== 0 \bmod s_2 \\ N+3 &== 0 \bmod s_3 \end{aligned}$$

or equivalently:

$$\begin{aligned} N &== 0 \bmod s_0 \\ N &== -1 \bmod s_1 \\ N &== -2 \bmod s_2 \\ N &== -3 \bmod s_3 \end{aligned}$$

(I'm using "==" to mean "congruent to," which is usually indicated by a symbol with three horizontal lines).

What are the values of N that satisfy these equations? The first equation means that $N = k_0 \cdot s_0$ for any integer k_0 . Plugging that into the second equation gives $k_0 \cdot s_0 = -1 \bmod s_1$. Can we simplify this into the form $k_0 = r \bmod s$? The answer is yes.

Solving modulo equations - there's an app for that!

In a more general sense, how do you solve $Ax == B \bmod C$? This is done with the [Linear Congruence Theorem](#). The procedure goes like this:

1. If $(B / \text{GCD}(A,C))$ not an integer, then there is no solution
2. Let $D = \text{GCD}(A,C)$.
3. Find r and s such that $rA + sC = D$
4. The solution is $x == (rB/D) \bmod (C/D)$

How do you find r and s? It's built-in to the 50G! The IEGCD command does it for you.

Here is the function that solves a single modulo equation:

```
@ This solves an equation of the form ax = b mod n
@ Where "=" really means "is congruent to".
@ It returns r s such that x = r mod s, or
@ equivalently x = s*k + r
@ arguments are a b n (must be integers)
@ Results are r s 1. if there's a solution, or
@ 0. if no solution exists.
@ See http://en.wikipedia.org/wiki/Linear_congruence_theorem
@ and the IEGCD command for details.
SLVMOD
\<< \-> a b n \<<
  @ If GCD(a,n) doesn't divide b evenly then no solution
  IF b a n GCD / DUP IP \=/ THEN
    0.
  ELSE
    @ It's good!
    a n IEGCD DROP
    b * OVER /
    n ROT /
    1. @ Indicates success
  END
\>>
\>>
```

Solving simultaneous modulo equations

Let's return to the original problem. We want to find N such that it simultaneously solves:

$$\begin{aligned} N &== 0 \bmod s_0 \\ N &== -1 \bmod s_1 \\ N &== -2 \bmod s_2 \\ N &== -3 \bmod s_3 \end{aligned}$$

More generally, suppose we want to solve two simultaneous equations:

```
X == b mod a, and
X == r mod s
```

we do it as follows:

```
X = a*k0 + b           (for any integer k0)
a*k0 + b == r mod s   {substituting for X into the second equation
k0*a == (r-b) mod s   {subtract b)
```

We can use SLVMOD on this equation to find:

```
k0 == u mod v, or equivalently
k0 = k1*v + u for any integer k1
```

and substituting for k0 into the equation above:

```
X = a*(k1*v+u) + b
  = (a*v)k1 + (a*u + b)
X == (a*u+b) mod (a*v)
```

The program SSLVMOD solves these simultaneous modulo equations:

```
@ Given two equations:
@ x = r mod s, and
@ x = b0 mod a0
@ This program returns b1 and a1 such that
@ x = b1 mod a1
@ Arguments: b0 a0 r s
@ results: b1 a1 1. (if there is a solution), or
@           0. if not
SSLVMOD
\<< \-> b a r s
\<< a r b - s
IF SLVMOD
THEN a * SWAP a * b + SWAP 1.
ELSE 0.
END
\>>
\>>
```

Putting it all together

The somewhat poorly named **FACTS program** takes a list of possible sums of digits {s0 s1 s2 s3} and computes a and b such that $N == a \bmod b$ and

```
N == 0 mod s0
N == -1 mod s1
N == -2 mod s2
N == -3 mod s3
```

Remember, this means that if {s0 s1 s2 s3} are a possible sum-of-digits, then the values $N == a \bmod b$ are the only possible numbers where N, N+1, N+2 and N+3 are evenly divisible by s0, s1, s2, and s3 respectively.

For example, given the list of possible sums-of-digits {4 5 6 7}, FACTS determines that $N == 4 \bmod 420$ are the only values that will work.

The **CHKSUM program** takes a list of possible digit sums and checks for a solution to the challenge that matches that list. In other words, it checks to see if there is a sequence of four 4-digit numbers that are (1) evenly divisible by the numbers in the list, AND (2) whose sum of digits equals the values in the list.

CHKSUM uses FACTS to determine the possible values that satisfy the first criteria. Then it uses the BIGGEST and SMALLEST arrays to look up the largest and smallest numbers that satisfy the sum-of-digits criteria. Finally, it uses a FOR loop to check each of the possible values to see if it matches the sum-of-digits criteria.

The **DIGITS program** takes an integer and computes the sum of its digits.

The **MAINP program** is the main entry point to the algorithm. It computes and stores the biggest and smallest 4-digit numbers for each possible sum. Then it calls SUMS to create the list of all possible sums of digits of sequences of 4-digit numbers. Finally, it uses DOLIST to call CHKSUM on each of these possible sums.

Conclusions

I think this is a good example of how you can squeeze performance out of a program by exploiting the properties of the problem that you're trying to solve. By starting with the potential sum of digits and then working forward to the possible 4-digit number (rather than the other way around), this program cuts down on the work needed:

1. Computing the biggest and smallest numbers with the given sum cuts the range of numbers that you need to check.
2. Computing the possible numbers that satisfy the divisibility test cuts the numbers down even more.

The program also demonstrates that it's worth stepping back from a problem and asking "is there a different approach that might work."

Program Listing

```
%%HP: T(3)A(R)F(.);
DIR
  MAINP
  \<< BIGandSMALL 'SMALLEST' STO 'BIGGEST' STO SUMS 1.
    \<< CHKSUM
    \>> DOLIST
  \>>
  DIGITS
  \<< 0. SWAP
  WHILE DUP
  REPEAT DUP 10. MOD ROT + SWAP 10. / IP
  END DROP
  \>>
  FACTS
  \<< 0. \-> L n
  \<< 0. 1. 1. L 1.
    \<<
    IF SWAP
    THEN n 'n' 1. STO- SWAP SSLVMOD
    ELSE DROP 0.
    END
  \>> DOLIST
  \>>
  CHKSUM
  \<< \-> L
  \<< L
  IF FACTS
  THEN L HEAD { } \-> a b N RES
  \<< SMALLEST N GET a - b / CEIL b * a + BIGGEST N GET
  IF DUP2 \<=
  THEN
  FOR I
  IF I DIGITS N ==
  THEN 1. 1. 3.
  FOR K I K + DIGITS L K 1. + GET == AND
  NEXT
  IF
  THEN I 'RES' STO+
  END
  END b
  STEP
  IF RES SIZE
  THEN RES
  END
  ELSE DROP DROP
  END
  \>>
  END
\>>
```

```
\>>
SUMS
\<< 1. 33.
  FOR i i DUP 1. + DUP 1. + DUP 1. + 4 \->LIST
  NEXT 8. 33.
  FOR i i DUP 1. + DUP 1. + i 6. - 4 \->LIST
  NEXT 9. 34.
  FOR i i DUP 1. + i 7. - i 6. - 4 \->LIST
  NEXT 10. 35.
  FOR i i DUP 8. - DUP 1. + DUP 1. + 4 \->LIST
  NEXT 19. 35.
  FOR i i DUP 17. - DUP 1. + DUP 1. + 4 \->LIST
  NEXT 18. 34.
  FOR i i DUP 1. + DUP 17. - DUP 1. + 4 \->LIST
  NEXT 17. 33.
  FOR i i DUP 1. + DUP 1. + DUP 17. - 4 \->LIST
  NEXT 28. 35.
  FOR i i DUP 26. - DUP 1. + DUP 1. + 4 \->LIST
  NEXT 27. 34.
  FOR i i DUP 1. + DUP 26. - DUP 1. + 4 \->LIST
  NEXT 26. 33.
  FOR i i DUP 1. + DUP 1. + DUP 26. - 4 \->LIST
  NEXT 186. \->LIST
\>>
BIGandSMALL
\<< 0. 36. NDUPN \->ARRY DUP \-> bigres smallres
  \<< 1. 36.
    FOR i 0. 1000. i \-> val mult cur
      \<<
        WHILE cur
          REPEAT cur 9. MIN 'cur' OVER STO- mult * 'val' STO+ 'mult' 10. STO/
          END 'bigres' i val PUT 1. 'mult' STO i 1. - 'cur' STO 1000. 'val' STO
          WHILE cur
            REPEAT cur 9. MIN 'cur' OVER STO- mult * 'val' STO+ 10. 'mult' STO*
            END 'smallres' i val PUT
          \>>
        NEXT bigres smallres
      \>>
    \>>
  SLVMOD
  \<< \-> a b n
    \<<
      IF b a n GCD / DUP IP \=/
      THEN 0.
      ELSE a n IEGCD DROP b * OVER / n ROT / 1.
      END
    \>>
  \>>
  SSLVMOD
  \<< \-> b a r s
    \<< a r b - s
      IF SLVMOD
      THEN a * SWAP a * b + SWAP 1.
      ELSE 0.
      END
    \>>
  \>>
END
```

Edited: 15 Mar 2012, 4:20 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

How to use a formula in a post

Posted by [Thomas Klemm](#) on 8 Nov 2010, 4:11 p.m.

Let's assume you want to use a formula in your post:

$$e^{i\pi} + 1 = 0$$

How can you do that?

1. Use the [equation editor](#) to enter the formula.
2. As a result you get the formula written in LaTeX:

```
e^{i\pi}+1=0
```

3. Choose *URL Encoded* at the bottom and copy it:

```
http://latex.codecogs.com/gif.latex?e%5E%7Bi%5Cpi%7D&plus;1%3D0
```

4. Insert the result in your post within an image-tag.

```
[img:http://latex.codecogs.com/gif.latex?e%5E%7Bi%5Cpi%7D&plus;1%3D0]
```

Egan Ford posted the following:

If you are too lazy to upload images you can use aamath for "a-text":

```
$ echo "e^(i*pi)+1=0" | aamath
```

```
ei π + 1 = 0
```

```
$ echo "sum([n-1;r],r = 0 .. min(n-1,4))" | aamath
```

```
min(n - 1, 4)
=====
\      / n - 1 \
/ >      |      |
/      \      /
=====
r = 0
```

Don Shepherd noted:

Quote:

Yeah, the MoHPC does have many of those symbols here: <http://www.hpmuseum.org/software/>

arrowl.gif	
delta.gif	
deltap.gif	
diamond.gif	
divide.gif	
dwnarrow.gif	
dwnquest.gif	
exch.gif	
exchi.gif	
gto.gif	
	
integral.gif	
interg.gif	
lesseq.gif	
meanx.gif	
noteq.gif	
notequal.gif	
pi.gif	
pict0.jpg	
rdelta.gif	
rolldn.gif	
rollup.gif	
rtbldarw.gif	
sigma.gif	
sqrt.gif	
stee.gif	
sum.gif	
summi.gif	
sumpl.gif	
symangle.gif	
symdel.gif	
symdelc.gif	
symgamma.gif	
	
symint.gif	
symnoteq.gif	
symphic.gif	
sympi.gif	
sympic.gif	
symsqrt.gif	
	
symsum.gif	
symsumm.gif	
symsump.gif	
symsums.gif	
symtheta.gif	
symthetc.gif	
uline.gif	
uparrow.gif	
yhat.gif	

Didier Lachieze proposed:

One way could be to use GIF/PNG images such as the ones [HERE](#).

For example: $\pi \infty \sum$

Edited: 18 Oct 2012, 4:43 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Sunrise and Sunset

Posted by **Thomas Klemm** on 28 Oct 2010, 12:05 p.m.

At the horticultural exhibition [Grün 80](#) I discovered this [precision sundial punctual to the minute](#) and still remember exactly how much I was impressed watching the time go by. It is still [there](#) if you ever happen to be in the vicinity.

Years later, I attended a lecture by [Christian Blatter](#): "[Von den Keplerschen Gesetzen zu einer minutengenauen Sonnenuhr](#)" (From Kepler's laws to an exact sundial). I'm sorry that the paper is only available in German but here's a similar paper in English: "[Another design for sundials](#)" by Norbert Hungerbühler. Though I knew about the [equation of time](#) and [Kepler's law](#) I wasn't aware of the interconnection until then.

Now that we can calculate the position of the sun for any date of the year I thought we could go one step further and calculate sunrise and sunset. However I was quiet disappointed by the result because it was very inaccurate compared to what I got for instance from <http://www.sunrisesunset.com>. It took me some time to understand why: It all depends on how you define sunrise and sunset. The usual value of the altitude of the center of the solar disc is about -0.83° while I assumed this to be zero. This may lead to an aberration of several minutes. On the other hand you're always late for the sunset if the upper edge just disappears.

The program is probably not the most precise. There are others around that do much better. My goal was to understand what's going on.

Best regards
Thomas

Formulae

From the paper I mentioned above the formula on page 158 was used in combination with the ansatz: $\psi(t) = t + d(t)$.

$$\psi(t) = t + 2 \sin t K + \frac{5}{4} \sin(2t) K^2$$

To make the calculation a bit easier I used the double-angle formula:

$$\sin(2t) = 2 \sin(t) \cos(t)$$

This leads to:

$$\psi(t) = t + K \sin t \left(2 + \frac{5}{2} K \cos t \right)$$

Both values $K \cos t$ and $K \sin t$ can be calculated in one step.

The expression in formula (6) on page 161 for mju(t) is stupendous. However it's far easier to use a suitable combination of polar/rectangular coordinate transformation. Don't tell me now you miss something on your HP-35s.

Constants:

- K = 0.016722 numerical orbital eccentricity of the earth
- E = 23.45 obliquity of the ecliptic
- P = 78.5 perihelion (relating to vernal equinox)
- U = 8:30'30" longitude (Zurich)
- V = 47:24'36" latitude (Zurich)

auxiliary quantity:

$$L^2 = \frac{1 + K}{1 - K}$$

Initialization:

(all angles must be stored in RAD)

- 0.016722 ST0 K
- 23.45 ->RAD ST0 E
- 78.5 ->RAD ST0 P
- 8.3030 ->HR ->RAD ST0 U
- 47.2436 ->HR ->RAD ST0 V
- 1 RCL+ K 1 RCL- K / SQRT ST0 L

Program for HP-32SII:

```
.....
: Main Program
:
: (dd.mm -> sunrise sunset)
:
.....
001 XEQ A
002 309
003 -
004 365
005 /
006 2
007 *
008 PI
009 *
010 XEQ B
011 XEQ C
012 GT0 D

.....
: Difference days
:
: (dd.mm -> days)
:
: calculates difference of days to March 1st
:
.....
A01 LBL A
A02 IP
A03 12
A04 LASTX
A05 FP
A06 100
A07 *
```

```
A08 3
A09 -
A10 x<0?
A11 +
A12 30.6
A13 *
A14 0.5
A15 +
A16 IP
A17 R^
A18 +
A19 RTN
```

```
.....
: Position of sun (approximation)
:
: (t -> psi)
:
:  $\psi(t) = t + 2 K \sin(t) + 5/4 K^2 \sin(2t) + \dots$ 
:
: .....
```

```
B01 LBL B
B02 ENTER
B03 ENTER
B04 RAD
B05 RCL K
B06 P->R
B07 5
B08 *
B09 2
B10 /
B11 2
B12 +
B13 *
B14 +
B15 RTN
```

```
.....
: Transformation to equatorial coordinates
:
: (psi -> mju delta)
:
:  $\phi = \psi - \alpha$ 
:
:  $x = \cos(\phi) = \cos(\delta) \cos(AR)$ 
:  $y = \sin(\phi) \cos(\epsilon) = \cos(\delta) \sin(AR)$ 
:  $z = \sin(\phi) \sin(\epsilon) = \sin(\delta)$ 
:
:  $mju = AR - (t - \alpha)$ 
:
: .....
```

```
C01 LBL C
C02 RCL- P
C03 RCL E
C04 x<>y
C05 1
C06 P->R
C07 RDN
C08 P->R
C09 R^
C10 R->P
C11 x<>y
C12 RDN
C13 R->P
C14 x<>y
C15 RDN
C16 RDN
C17 RCL- P
C18 -
C19 RTN
```

```
.....
: Sunrise and Sunset
:
: (mju delta -> Sunrise Sunset)
:
: .....
```

```
D01 LBL D
D02 RCL+ U
D03 ENTER
```

```
D04 ENTER
D05 R^
D06 TAN
D07 RCL V
D08 TAN
D09 *
D10 ACOS
D11 +
D12 x<>y
D13 LASTx
D14 -
D15 XEQ E
D16 x<>y
```

```
.....
: Angle to Hours
:
: (RAD -> HMS)
.....
```

```
E01 LBL E
E02 2
E03 PI
E04 *
E05 /
E06 FP
E07 1
E08 +
E09 FP
E10 24
E11 *
E12 ->HMS
E13 RTN
```

```
.....
: Test approximative solution of Kepler Equation
:
: (psi -> t)
.....
```

```
F01 LBL F
F02 2
F03 /
F04 TAN
F05 RCL/ L
F06 ATAN
F07 2
F08 *
F09 ENTER
F10 SIN
F11 RCL* K
F12 -
F13 RTN
```

Let me add a few things:

- Should you be more interested in the equation of time just remove the call to subroutine **D** in the main program. The result is in RAD and may be transformed to hours using subroutine **E**.
- The subroutine **F** isn't really needed. It may be used to check the approximation used in subroutine **B**. These two programs are (not exactly) inverse to each other.
- I organized the program into smaller modules that can be used and tested independently. However the program can be unfolded to save some space.
- In the listing below I've added the status of the stack for each step to ease reading the program.
- Initially the program was written for the HP-11C but I preferred using letters for the registers. That's why I have selected the HP-32SII. However it's straightforward to transform that program to the HP-41 or HP-15C.

Annotated Listing

L		X	Y	Z	T
dd.mm		dd.mm			
001	IP	d			
002	12	12	d		
003	LASTX	dd.mm	12	d	
004	FP	0.mm	12	d	
005	100	100	0.mm	12	d
006	*	m	12	d	d
007	3	3	m	12	d
008	-	m - 3	12	d	d
009	x<0?	m - 3	12	d	d
010	+	n	?	d	d
011	30.6	30.6	n	?	d
012	*	30.6 n	?	d	d
013	0.5	0.5	30.6 n	?	d
014	+	30.6 n + 0.5	?	d	d
015	IP	D	?	d	d
016	R^	d	D		
017	+	days			
018	309	309	days		
019	-	days - 309			
020	365	365	days - 309		
021	/	T			
022	2	2	T		
023	*	2 T			
024	PI	pi	2 T		
025	*	t			
026	ENTER	t	t		
027	ENTER	t	t	t	
028	RAD	t	t	t	
029	RCL K	K	t	t	t
030	P->R	K cos t	K sin t	t	t
031	5	5	K cos t	K sin t	t
032	*	5 K cos t	K sin t	t	t
033	2	2	5 K cos t	K sin t	t
034	/	5/2 K cos t	K sin t	t	t
035	2	2	5/2 K cos t	K sin t	t
036	+	2 + 5/2 K cos t	K sin t	t	t

037 *	d(t)	t	t	t
038 +	psi	t	t	t
039 RCL - P	phi	t	t	t
040 RCL E	eps	phi	t	t
041 x<>y	phi	eps	t	t
042 1	1	phi	eps	t
043 P->R	x	r	eps	t
044 RDN	r	eps	t	x
045 P->R	y	z	t	x
046 R^	x	y	z	t
047 R->P	r	AR	z	t
048 x<>y	AR	r	z	t
049 RDN	r	z	t	AR
050 R->P	1	delta	t	AR
051 x<>y	delta	1	t	AR
052 RDN	1	t	AR	delta
053 RDN	t	AR	delta	
054 RCL - P	t - alpha	AR	delta	
055 -	mju	delta		
056 RCL+ U	tau	delta		
057 ENTER	tau	tau	delta	
058 ENTER	tau	tau	tau	delta
059 R^	delta	tau	tau	tau
060 TAN	tan delta	tau	tau	tau
061 RCL V	theta	tan delta	tau	tau
062 TAN	tan theta	tan delta	tau	tau
063 *	cos omega	tau	tau	tau
064 ACOS	omega	tau	tau	tau
065 +	rho	tau	tau	tau
066 x<>y ^{omega}	tau	rho	tau	tau
067 LASTx ^{omega}	omega	tau	rho	
068 -	sigma	rho		
069 XEQ E	set	rho		
070 x<>y	rho	set		
E01 LBL E	rho	set		
E02 2	2	rho	set	
E03 PI	pi	2	rho	set
E04 *	2 pi	rho	set	
E05 /	r	set		

E06	FP	[r]	set	
E07	1	1	[r]	set
E08	+	r'	set	
E09	FP	[r']	set	
E10	24	24	[r']	set
E11	*	r''	set	
E12	->HMS	mm.hhss	set	
E13	RTN	rise	set	

Quote:

it's straightforward to transform that program to the HP-41

Registers:

- 00 K = 0.016722 numerical orbital eccentricity of the earth
- 01 E = 23.45 obliquity of the ecliptic
- 02 P = 78.5 perihelion (relating to vernal equinox)
- 03 U = 8:30'30" longitude (Zurich)
- 04 V = 47:24'36" latitude (Zurich)

Program for HP-41C:

```
01 LBL "SUN"
02 INT
03 12
04 LASTX
05 FRC
06 100
07 *
08 3
09 -
10 X<0?
11 +
12 30.6
13 *
14 0.5
15 +
16 INT
17 R^
18 +
19 309
20 -
21 365
22 /
23 2
24 *
25 PI
26 *
27 ENTER
28 ENTER
29 RAD
30 RCL 00

31 P-R
32 5
33 *
34 2
35 /
36 2
37 +
38 *
39 +
40 RCL 02
41 -
42 RCL 01
43 X<>Y
44 1
45 P-R
46 RDN
47 P-R
48 R^
49 R-P
50 X<>Y
51 RDN
52 R-P
53 X<>Y
54 RDN
55 RDN
56 RCL 02
57 -
58 -
59 RCL 03
60 +

61 ENTER
62 ENTER
63 R^
64 TAN
65 RCL 04
66 TAN
67 *
68 ACOS
69 +
70 X<>Y
71 LASTX
72 -
73 XEQ 00
74 X<>Y
75 LBL 00
76 2
77 PI
78 *
79 /
80 FRC
81 1
82 +
83 FRC
84 24
85 *
86 HMS
87 END
```

Appendix:

Earth's perihelion and aphelion

Year	Perihelion		Aphelion	
	Date	Hour	Date	Hour
2007	January 3	20:00	July 7	00:00
2008	January 3	00:00	July 4	08:00
2009	January 4	15:00	July 4	02:00
2010	January 3	00:00	July 6	11:00
2011	January 3	19:00	July 4	15:00
2012	January 5	00:00	July 5	03:00
2013	January 2	05:00	July 5	15:00
2014	January 4	12:00	July 4	00:00
2015	January 4	07:00	July 6	19:00
2016	January 2	23:00	July 4	16:00
2017	January 4	14:00	July 3	20:00
2018	January 3	06:00	July 6	17:00
2019	January 3	05:00	July 4	22:00
2020	January 5	08:00	July 4	12:00

Orbital Paramameters for the Earth

Year (A.D.)	Eccentri city	Obliquity (degrees)	Long. of Perihel. (degrees)
1990	.016708	23.4411	282.724
1991	.016707	23.4409	282.741
1992	.016707	23.4408	282.758
1993	.016707	23.4407	282.776
1994	.016706	23.4405	282.793
1995	.016706	23.4404	282.810
1996	.016705	23.4403	282.827
1997	.016705	23.4402	282.844
1998	.016704	23.4400	282.861
1999	.016704	23.4399	282.878
2000	.016704	23.4398	282.895
2001	.016703	23.4396	282.913
2002	.016703	23.4395	282.930
2003	.016702	23.4394	282.947
2004	.016702	23.4392	282.964
2005	.016702	23.4391	282.981
2006	.016701	23.4390	282.998
2007	.016701	23.4389	283.015
2008	.016700	23.4387	283.033
2009	.016700	23.4386	283.050
2010	.016700	23.4385	283.067
2011	.016699	23.4383	283.084
2012	.016699	23.4382	283.101
2013	.016698	23.4381	283.118
2014	.016698	23.4379	283.135
2015	.016698	23.4378	283.152
2016	.016697	23.4377	283.170
2017	.016697	23.4376	283.187
2018	.016696	23.4374	283.204
2019	.016696	23.4373	283.221
2020	.016696	23.4372	283.238

Edited: 1 Nov 2010, 10:20 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

HP 35s. Quadratic Equation Solver (with complex solutions support)

Posted by **Pablo P (Spain)** on 2 July 2010, 3:53 p.m.

HP 35s - QUADRATIC EQUATION SOLVER (WITH COMPLEX SOLUTIONS SUPPORT)

Table of Contents

=====

1 Using the program

2 Program listing

3 Registers

4 Revision

1 Using the program

~~~~~

Execute the program: XEQ Q ENTER.  
Enter A, B and C according to the equation  $Ax^2+Bx+C=0$ .  
It will show both solutions in x and y registers.

2 Program listing

~~~~~

Line	Instruction
-----+-----	
Q001	LBL Q
Q002	CF 10
Q003	INPUT A
Q004	INPUT B
Q005	INPUT C
Q006	-B÷(2xA)
Q007	ENTER
Q008	ENTER
Q009	B^2-4xAxC
Q010	x>=0?
Q011	GTO Q017
Q012	+/-
Q013	sqrt ;press k key
Q014	i
Q015	x
Q016	GTO Q018
Q017	sqrt
Q018	2xA
Q019	÷
Q020	+
Q021	x<>y
Q022	LASTx
Q023	-
Q024	RTN

93 Bytes

3 Registers

~~~~~

| Register | Use             |
|----------|-----------------|
| -----+   |                 |
| A        | x^2 coefficient |
| B        | x^1 coefficient |
| C        | x^0 coefficient |

4 Revision  
~~~~~

05/18/10 - Initial release.
06/25/10 - Smaller program, D E F variebles not used.

Edited: 2 July 2010, 3:56 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Calculator Benchmark: Loops of addition

Posted by [gene wright](#) on 2 June 2010, 2:51 p.m.

Presently in order from fastest to slowest.

This benchmark goes after ONLY goes after one thing: loops of addition.

LBL 01 + GTO 01

where the stack contains 0 1 1 1 on 4 level machines. In other words, start with 0 and do repeated additions of 1 for 60 seconds. What is the result? Alternative methods that count 1 at a time are allowed and included, such as going through an ISG / DSE loop. Long repeated programs of + + + + are not.

The list shows:

The machine used The count after 60 seconds The program code.

=====

```
Casio fx-9860g Slim Compiled C program SDK V 1.0
Count: 349,700,000
Code:
int AddIn_main(int isAppli, unsigned short OptionNum)
{
  unsigned int key;
  unsigned long int counter = 0;
  Bdisp_AllClr_DDVRAM();
  do {
    counter++;
  } while (counter < 349700000);
  locate(1,5);
  Print((unsigned char*)" end");
  while(1){
    GetKey(&key);
  }

  return 1;
}
```

```
HP-12C+, Scott's custom integer firmware, overclocked to 48.75 MHz
Count: 261,602,459
Code: Unspecified
```

```
HP 50g 2.15, HPGCC, 75MHz
Count: 161,722,281
Code:
#include <hpgcc49.h>
int main(void)
{
  unsigned int c=0;
  unsigned int volatile * GPFDAT = (int*) 0x7A00054;
```

```

sys_slowOff(); //press ON to stop
while(!(*GPFDAT&1))
    c++;
sat_push_zint_llong(c);
return 0;
}

```

HP-12C+, Scott's custom firmware

Count: 10,794,647

Code:
Do { ++x; } while (((*(unsigned long int*)PIOC_PDSR) & c_mask)==c_mask));

HP-71B USA:2504A00223 HP71:1BBBB FTH:1A, year of production 1984?

Count: 1,069,543

Code: Forth + Assembler: ADD1 D.

Press any key (except ON) to stop counting.

Count => double length value on the stack.

First assemble ADD1 into the Forth dictionary:

```

File: ADD1
FORTH
WORD 'ADD1'
** overflows in about 470 secs .... P= xx where xx>6 gives more time
P= 6
C=0 A
A=0 WP
LOOP A=A+1 WP
C=IN // Keyboard touched (except ON key !)
?C=0 A
GOYES LOOP
D1=D1- 5
DAT1=A A
ASR W
ASR W
ASR W
ASR W
ASR W
D1=D1- 5
DAT1=A A
RTNCC
END

```

BR

HP 30b with TSTSYS ON

Count: 531,131

Code: LBL 00 ISG 0 Stop GOTO 00, with 0 pre-stored in 0. Result indicates count of loops in one minute. Each loop adds 1 to value in memory 0.
(as first program in memory)

FX-9860G SD Fast Mode

Count: 370,230

Code: Lbl 0 : Isz A : Goto 0

HP 34s Build 1099

Count: 272,538

Code: LBL A, +, BACK 01

FX-9860G SD Fast Mode

Count: 237,780

Code: Lbl 0 : Ans+1 : Goto 0

HP 30b with TSTSYS ON

Count: 222,578

Code: LBL 00 + GOTO 00
(as first program in memory)

HP 30b with TSTSYS ON
Count: 209,038
Code: LBL 00 STO+0 GOTO 00 with 0 pre-stored in 0 and 1's filling the stack
(as first program in memory)

FX-9860G SD Fast Mode
Count: 206,250
Code: 0->A : Lbl 0 : A+1->A : Goto 0

FX-9860G SD
Count: 177,230
Code: Lbl 0 : Isz A : Goto 0

HP 50g (SysRPL bint)
Count: 124,445
Code: !NO CODE !RPL :: ZERO BEGIN #1+ GETTOUCH UNTIL DROP ; @

FX-9860G SD
Count: 110,140
Code: Lbl 0 : Ans+1 : Goto 0

FX-9860G SD
Count: 97,950
Code: 0->A : Lbl 0 : A+1->A : Goto 0

HP 12c+
Count: 78,640
Code: + GT0 01

HP 30b
Count: 72,517
Code: LBL 00 + GOTO 00
(as first program in memory)

HP 9825 B
Count: 59,568
Code: 0->A; "start"; A+1->A; gto "start"; end

HP 50g (SysRPL floating point)
Count: 56,994
Code: !NO CODE !RPL :: BEGIN %1+ GETTOUCH UNTIL DROP ; @

Casio fx-9860G Slim
Count: 55,924
Code: 0->A Lbl 0 A+1->A Goto 0

HP-9100A SERIAL NUMBER 816-01071 (1971?)
Count: 49,391
Code: +, GO TO 00

HP 9815S
Count: 47,592
Code: 000 + 001 GOTO 000

HP-9100B
Count: 47,394
Code: +, GO TO, 0, 0

HP-71B Forth Xerxes:
COUNT: 46,575
Code:
Forth
: XERXES 1 SWAP 0 DO 1+ LOOP ;

46575 XERXES DROP
BYE

HP 50g
Count: 31,849
Code: 1. << DO 1. + UNTIL 0. END >>

HP 9815A/S
Count: 31,156
Code; LBL, A, 1, +, GTO A

HP 48gII Exact mode (old version)
Count: 28,160
Code: 1. << DO 1. + UNTIL 0. END >> EVAL

HP 48gII
Count: 26,439 // Approx Mode
Code: << WHILE 1 REPEAT 1. + END >>

TI-83+ SE
Count: 24,840
Code: Lbl 0 : Ans+1 : Goto 0

HP 85
Count: 23,605
Code: 10 Let A=0; 20 Let A=A+1; 30 goto 20; 40 END

Sharp EL-9900
Count: 23,483
Code: Label X A+1=>A Goto X

HP-75C (1982)
Count: 21,717
Code: 10 A=0 20 A=A+1 @ GOTO 20 30 END

TI-81
Count: 20,191
Code: Lbl A : Ans + 1 : Goto A : End

Casio FX-7500G
Count: 15,364
Code: Lbl 1 Isz A Goto 1

HP 9810A
Count: 15,355
Code LBL, 1, +, GTO, 1

TI-81
Count: 15,116
Code: Lbl A : B + 1 > B : Goto A : End

TI-84 Silver Edition
Count: 13,838
Code: Lbl 1:A+1->A: Goto 1

TI-83
Count: 12,986
Code: Lbl B: ans +1: goto B

HP 49G
Count: 12,351
Code: 1. << DO 1. + UNTIL 0. END >>

The NEWT 41CL logic board upgrade for the HP 41. Code: LBL 00 + GTO 00

1X mode: 1,055
2X Turbo mode: 1,913
5X Turbo mode: 4,153
10X Turbo mode: 6,538
20X Turbo mode: 9,179
50X Turbo mode: 12,022

HP 48GX
Count: 11,636
Code: << WHILE 1 REPEAT 1 + END >>

TI-85
Count 11,100
Code: Lbl A : Ans + 1 : Goto A : End

HP 49G
Count: 11,041
Code: << WHILE 1. REPEAT 1. + END >>

Psion Organiser II CM (1997)
Count: 10,949
Code:  LOCAL a a=1 D0 a=a+1 UNTIL KEY\$="S" PRINT a GET

HP-48G
Count: 10,726
Code: << WHILE 1 REPEAT 1 + END >>

Casio FX-7500G
Count: 10,692
Code: Lbl 1 Ans+1 Goto 1

TI-83+
Count 10,385
Code: Lbl A : Ans + 1 : Goto A : End

HP 33s
Count: 10,097
Code: LBL A + GTO A

TI CC-40
Count: 9,339
Code: 10 A = A+ 1 : 20 GOTO 10

TI-89 Titanium
Count: 9,339
Program: aa():Prgm:Lbl b:a+1->a:Goto 1: EndPrgm

Casio FX-7500G
CountL 8,878
Code: Lbl 1 A+1->A Goto 1

TI-83
Count: 8,106
Code: LBL B, A+1>>A, goto B

Casio FX-7500G
Count: 8,105
Code: 0->A Lbl 1 A+1->A Goto 1

Casio fx-7000GB
Result: 7,437
Code: Lbl 1; A+1->A; Goto A

HP-48SX
Count: 7,352
Code: Unknown

FX-603P
Count: 7,240
Code: AC 1 + + LBL 0 = GOTO 0

HP-42s FAST MODE S/N 2849A with goose disabled
Count: 6,485
Code: CLLCD LBL 01 + GT0 01

HP-32s
Count: 5,973
Code: LBL A + GT0 A

TI-92
Count: 5,686
Code: 0->a Lbl aa a+1->a Goto aa

HP-28S
Count: 5,677
Code: HOME 1 << WHILE 1 REPEAT 1 + END >> EVAL

HP 50G (normal speed)
Count: 5,510
Code: << 1 + A >> stored in A, start with 1 on stack line 1:

HP 35s
Count: 5,504
Code: B001 LBL B B002 STO+ Z B003 GT0 B002, with Z initialized to 0

TI-86
Count 5,347
Code: Lbl A : Ans + 1 : Goto A : End

FX-5800P
Count: 5,340
Code: Lbl 0 : Ans+1 : Goto 0

PC-1247
Count: 5,180
Code: 1:A=A+1:GOTO1

HP 20S
Count: 4,837
Code: LBL A, +, GT0 A

HP-32sii
Count: 4,715
Code: LBL A, +, GT0 A

HP-42s FAST MODE S/N 2849A
Count: 4,419
Code: LBL 01 + GT0 01

HP-71B
Count: 4,320
Code:

10 DESTROY A @ A=0
20 ON TIMER #1,60 GOTO 40
30 A=A+1 @ GOTO 30

40 DISP A

HP 48gII (Original 3 batteries, NO USB)

Count: 4,296

Code: << TICKS 8192 60 * + << -> t << DO 1. + UNTIL TICKS t >= END >> >> EVAL >>

HP-20S

Count: 4,170

Code: Unknown

HP 35s

Count: 3,652

Code: LBL B, +, GTO B001

TI-86

Count: 3,612

Code: Lbl B:1+A->A:Goto B

HP 39gs

Count: 3,607

Code: 1->A: DO A+1->A UNTIL A<=0 END:

Aurora HP 12c clone

Count: 3,554

Code: +, GTO 01

TI-95

Count: 3,350 counts

Code: 1 + GTO 0000

Radio Shack PC-2

Count: 3,317

Code: 1: A=A+1 2: GOTO 1 ; start with 0 stored in A.

Casio fx602p

Count: 3,111

Code: AC 1++ LBL1 = GOTO1

Sharp EL-9650

Count: 3,093

Code: Label X A+1=>A Goto X

HP 48gII (Original 3 batteries, NO USB)

Count: 3,088

Code: << WHILE 1 REPEAT 1. + END >>

HP-42S with "goose" disabled. Non-fast mode.

Count: 3,067

Code: CLLCD, LBL 00 + GTO 00

HP 48gii Exact mode (Original 3 batteries, NO USB)

Count: 3,036

Code: << WHILE 1 REPEAT 1 + END >>

TI-80

Count 2,279

Code: Lbl A : Ans + 1 : Goto A : End

HP-42S

Count: 2,115

Code: LBL 00 + GTO 00

Sharp EL-5500III

Count: 2,056
Code: 1 A=A+1: GOTO 1

HP-41CY Turbo
Count: 1,982
Code: LBL 00, +, GTO 00

Sharp EL-5500II
Count: 1,510
Code: 1 A=A+1: GOTO 1

HP 9G
Count: 1,470
Code: A=0; Lbl 0=; A=A+1; GOTO 0; END

HP 12c plat 25th anniv.
Count: 1,435
Code: +, Goto 001

HP 12c platinum
Count: 1,386
Code: + GTO 001

HP41CX Synthetic:
Count: 1,298
Code:
01 LBL "X"
02 1
03 "SeeRemark"

!!! Synthetic string 9 long Decimal 249 96 7 117 131 131 131 64 178 176
this places 96 7 in N to be transferred in reg b, resulting in
program counter in reg M, byte 6. In reg M synthetic code results
in: RDN ENTER ENTER ENTER + GTO 01. The GTO 01 is a COMPILED GTO
jumping to the + before it! You can SST this code but the GTO 01
takes a long time to reposition! Yes, this can be optimized still a
bit, but the loop is using 3 bytes!

04 0
05 X<> N
06 BEEP
07 STOP
08 STO b
09 END

INSTRUCTION: READ THIS
XEQ "X"
wait until beep, 000000,06 is displayed
press [R/S] to start counting
press [R/S] after 60 seconds, write down the count
perform a GTO .. !!! because the program pointer is in reg M, switching to program mode and
TOUCHING A KEY results in MEMORY LOST!

CASIO PB-700
Count: 1,282
Code: 1: A=A+1 2: GOTO 1 ; start with 0 stored in A.

Sharp PC-1251
Count: 1,277
Code: 1: A=A+1 2: GOTO 1 ; start with 0 stored in A.

HP41CX:
Count 1,075
Code: LBL 01 + GTO 01

HP 41c

Count: 1,063
Code: LBL 01 + GT0 01

TI-59
Count: 635
Code: 1 + RST

HP-65
Count: 578
Code: Lbl 1, +, Goto 1

HP-10C
Count: 514
Code: LBL 0 + GT0 0

HP-25
Count: 512
Code: 01 + 02 GT0 01

TI-59 starting with 1 in the display
Count: 501
Code: SUM 00 RST

HP 12c original (mfg. 1985)
Count: 500
Code: +; GT0 01

TI-57
Count: 498
Code: + 1 RST

TI-59
Count: 492
Code: OP20 RST

HP 25
Count: 483
Code: + GT0 01

TI SR-56
Count: 461
Code: + 1 = RST

TI SR-52
Count: 425
Code: + 1 = RST

HP-38C
Count: 387
Code: + GT0 01

TI 58C
Count: 387
Code: + 1 = RST

HP-29C
Count: 361
Code: LBL 0; +; GOTO 0

SHARP PC-1211
Count: 358
Code 1:Z=Z+1:GOTO 1 (starting with Z=0)

HP 11c
Count: 353
Code: Lbl 1, +, Goto 1

TI-59
Count: 336
Code: 1 + GT0 000

HP-33c (mfg. 1982)
Count: 319
Code: 01 + 02 GT0 01

HP 67
Count: 349
Code: 1 CHS STO I + GOTO (i)

HP 55
Count: 312
Code: + GT0 01

HP-19c
Count: 301
Code: LBL 1 + GT0 1

HP 15c (mfg. 1983)
Count: 297
Code: LBL A; +; GT0 A

HP 16c: 293 (float 4)
Count: 293
Code: LBL A + GT0 A

HP 34c: (I program) with -1 stored in I
Count: 277
Code: LBL A + GT0 f I

HP 34c (mfg. 1981)
Count: 269
Code: LBL A; +; GT0 A

TI-55
Count: 267
Code: + 1 RST

HP 34c:
Count: 259
Code: LBL A + GT0 A

HP 16c
Count: 245
Code: Lbl 1, +, Goto 1

HP 16c: 230 (decimal, wsize 16, 2-complement)
Count: 230
Code: LBL A + GT0 A

HP 67
Count: 226
Code: LBL 1 + GT0 1

HP-97
Count: 223
Code: LBL1; +; GOTO 1

Texas Instrument TI-57LCD
Count: 216
Code: LBL_01 1 + GTO_01

TI-66
Count: 210
Code: + 1 = RST

TI-65
Count: 205
Code: 1 + RST

Texas Instrument TI-57LCD
Count: 195
Code: 1 + RST

TI-55
Count: 189
Code: + 1 = RST

TI BA 55
Count: 139
Code: + 1 = RST

Human hand Tapping + key
Count: 133

Elektronika MK-61
Count: 106
Code: + GSB 00

TI-62
Count: 100
Code: 1 + RST

Commodore P50
Count: 97
Code: + 1 = SKZ GOTO 00 R/S (Start with -97)

TI-59 with RPN Module:
Count: 74
HP-67 code: LBL 1 + 1 GOTO 1
The RPN Module translated that to:
LBL LNX PGM 51 A 1 PGM 12 A GTO LNX

Edited: 13 June 2011, 3:11 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

The Caves - Reloaded

Posted by [Frido Bohn](#) on 6 May 2010, 5:03 a.m.

The Caves is a classic HP41-game designed by James Surber and published in the Games II volume of the Users' Library Solution. Its listing can be found at this [site](#).

Game Summary

This is a typical maze game with 67 interconnected caves. The main goal is to collect the treasures to an amount of 1200 points. The caves contain also several hazards and some gadgets.

The game starts asking for a seed number. Thereof the items of the game will be distributed across all caves. Each seed number will generate a specific distribution pattern which can be restored using the same number again in a later game.

Some seconds after entering the seed number, you enter the game in Cave 0 in user mode and will be informed whether some hazards exist in the adjacent caves or not, but you will not be informed in what particular cave the hazard is. The only means to get more information about the caves is a wizard (hear voice) who will tell you something about the five lower and upper caves as seen from his cave. Otherwise you have to adventure the next caves by luck and logical deductions.

Each cave is connected to three other caves, with two exceptions, with only two neighbouring caves. The leftmost number in the display is followed by an "*", and marks the cave where you are just in. The other three comma-separated numbers display the adjacent caves. By entering the respective cave number you will get access to that cave. There are also stairs and forks. Stairs will give you the option to go up- [Key F] or downstairs [Key G] to a random lower- or higher-numbered cave – thus, you have no control where you may end. The forks give you the option to go right [Key I] or left [Key H]. In this situation, the number of the caves will be shown to you, thus giving you a bit more control. Finally, you may go backwards to the cave you just left [Key J].

Your basic equipment consists of five tools (knife, water, gloves, lamp and keys) and may be expanded by a wand and a cape if you are lucky to enter the respective caves. Part of the equipment will defend you against dragons (cape), fire (water), trolls (knife), dark caves (lamp) or grant you access through locked doors (keys). Other equipment is useful to collect the treasures, i.e. gloves for gold (AU) and silver (AG), as well as for rubies and silk which are locked in chests for which the keys are to be used.

Some hazards like holes (draft) and a kind of waterfalls (splash) irrevocably kick you out of the game. Each time, you enter a new cave you will be informed about the content of the neighbouring caves, with few exceptions (trolls, fire, locked doors, web, dark cave, small cave). Most of them can be handled with the standard equipment. But standard equipment may be lost if you enter caves containing bats (hear squeaks). In such a case you may conjure them back if you use the wand. The wand itself must be found first in a cave, and will work twice only after each visit to that cave. Likewise with the cape. It is useful to hide oneself when entering caves containing dragons (smell dragon), but it can be lost if you get into small caves. Of course you can adventure yourself without some standard equipment but this could be risky as you might enter a cave with a troll without warning and without your knife which finally would throw you off the game. A cave with fire would burn you if you have no water, and a cave with a web or a locked door would deny you the access if you have no knife to cut the web or keys to open the door.

Another gadget to be found in a cave is the magic word. Upon using it will get you out off the maze in healthy conditions. The other way of a healthy escape is the use of an exit-cave (see light).

I hope this brief description gave you some clue about the aim of the game. As stated above, the game is introduced in detail in the Games II Users' Library Solution book.

System Requirements

For an HP41 it is challenging because all 319 registers are needed. Thus, it can only be handled by a 41C with full memory, or of course by the CV or CX. But in either configuration, you will not have enough room for any other programs. It is recommended to use a fresh (memory lost) system and start with zero registers (SIZE 000) in order to accommodate the program first. After packing, you will have to give the game 76 registers (SIZE 076) in order to start it properly. For the best performance I advice to delete the key assignments before starting the game to have full access to the controls of the game which uses labels A to J.

User Requirements

The game is not only challenging for the HP41. This applies also to oneself. When I started to play the game some years ago, I found that my mnemonic capabilities where driven to its limits. First, it is difficult to understand the system of the interconnections between the caves. Second, you have to deduce where the hazards might be, as you only will get hints about their rough location.

Therefore, I recently had the idea to prepare a map which will ease the path through the cave-system. If printed out, you may use it for your notes. The location of hazards can be nailed if you by-pass the suspected cave through another cave in the maze. The map is also good to annotate the hints of the wizards.

Additionally, I gave a basic scheme of the hazards, its remedies, the HP41-key assignments, and scores of the treasures. The lines between the caves display their interconnections. Do not be confused if they are dash-dotted or full, this was only done to mark the correct path. The paths from Cave 7 to Cave 0, and from Cave 60 to Cave 61 are one-way only. On the right lower edge, you will find "Seed" where you can annotate the seed number you used at the initiation of the maze. Very useful, if you want to re-challenge that nasty dragon which kicked you off the game. A tip: if you want to get rid of the sound, clear flag 26 [CF 26].

Conclusion

Although a game of the computer stone-age it is quite fun to play and demands the use of neurons which usually are left in stand-by. All you need is an HP41, a sheet of paper and a pencil. I may quote my 8 year old son who stated that the game is "somewhat more challenging" than most of his hand held console programs.

Note: The CAVES map may be obtained directly from me (fridboh at yahoo.co.uk).

Edited: 30 May 2012, 11:40 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Table of HP48 serial numbers and ROM versions

Posted by [megarat](#) on 13 Apr 2010, 12:22 p.m.

I'm compiling serial number and ROM information of HP48 calculators with the intent of determining their relationship. The table is below, and the relationship is pretty apparent.

I've also parsed the information from the serial number (country, week, year) to make it slightly more human-readable. And heck, it might come in handy if you want to load this into your favorite data-mining program or something.

Thanks to all who have submitted their information. Additional contributions would be appreciated, so feel free to join in by contacting me through the web site, especially if your calculator helps define the boundary conditions (i.e., where the ROM versions change).

For convenience, here's a quick primer on checking your version number (the process is model-dependent):

-- 48g family: type the command 'VERSION'

-- 48s family: hold ON and D simultaneously; release D; then release ON; then hit backspace, then hit EVAL.

Behold, the table:

Model	ROM	SerialNum	Country	Year	Week	Display	Keys
48sx	A	2952A00211	USA	1989	52		
48sx	A	3003A00504	USA	1990	03		
48sx	B	3013A01447	USA	1990	13		
48sx	C	3016A00995	USA	1990	16		
48sx	C	3017A01085	USA	1990	17		
48sx	D	3021A00931	USA	1990	21		
48sx	D	3027A01190	USA	1990	27		
48sx	D	3028A00869	USA	1990	28		
48sx	D	3028A02420	USA	1990	28		
48sx	D	3029A02006	USA	1990	29		
48sx	D	3030A02142	USA	1990	30		
48sx	D	3034A01410	USA	1990	34		
48sx	E	3038A03578	USA	1990	38		
48sx	E	3052A01149	USA	1990	52		
48sx	E	3103A02201	USA	1991	03		
48sx	E	3104A02185	USA	1991	04		
48sx	E	3106A00196	USA	1991	06		
48sx	E	3106A02095	USA	1991	06		
48sx	E	3107A01152	USA	1991	07		
48sx	E	3111A00584	USA	1991	11		
48sx	E	3112A00435	USA	1991	12		
48sx	E	3115A00495	USA	1991	15		
48sx	E	3116S01777	Singapore	1991	16		
48sx	E	3117A00013	USA	1991	17		
48sx	E	3117S00252	Singapore	1991	17		
48s	E	3119S02543	Singapore	1991	19		
48sx	E	3132S00211	Singapore	1991	32		
48sx	E	3137S03581	Singapore	1991	37		
48sx	E	3142S02622	Singapore	1991	42		
48sx	E	3152S03955	Singapore	1991	52		
48sx	E	3204S03206	Singapore	1992	04		
48sx	E	3208S02217	Singapore	1992	08		

Table of HP48 serial numbers and ROM versions

48sx	E	3208S05906	Singapore	1992	08	
48s	E	3214S00428	Singapore	1992	14	
48sx	E	3217S00646	Singapore	1992	17	
48sx	E	3217S00674	Singapore	1992	17	
48s	E	3225S04049	Singapore	1992	25	
48s	E	3240S00388	Singapore	1992	40	
48s	E	3242S02348	Singapore	1992	42	
48s	J	3246S07047	Singapore	1992	46	
48sx	J	3247S01125	Singapore	1992	47	
48s	J	3301S02361	Singapore	1993	01	
48s	J	3305S05983	Singapore	1993	05	
48sx	J	3310S04480	Singapore	1993	10	
48s	J	3311S00452	Singapore	1993	11	
48sx	J	3320S00496	Singapore	1993	20	
48g	L	3323S01820	Singapore	1993	23	
48gx	M	3324S05822	Singapore	1993	24	Normal
48g	M	3325S04787	Singapore	1993	25	
48gx	M	3327S02483	Singapore	1993	27	
48g	M	3344S06734	Singapore	1993	44	
48g	P	3348S06483	Singapore	1993	48	
48gx	R	3403S02759	Singapore	1994	03	Normal
48g	R	3421S05184	Singapore	1994	21	Normal
48g	R	3422S07894	Singapore	1994	22	
48gx	R	3433S02345	Singapore	1994	33	
48gx	R	3544S08544	Singapore	1995	44	
48g	R	3610S03702	Singapore	1996	10	
48gx	R	3616S02571	Singapore	1996	16	
48gx	R	3627S01702	Singapore	1996	27	
48gx	R	3629S01544	Singapore	1996	29	
48gx	R	3705S08111	Singapore	1997	05	
48gx	R	SG72103103	Singapore	1997	21	Normal
48gx	R	SG72400781	Singapore	1997	24	
48gx	R	SG72703278	Singapore	1997	27	
48gx	R	ID73607337	Indonesia	1997	36	
48g+	R	ID83904475	Indonesia	1998	39	
48g+	R	ID94300150	Indonesia	1999	43	
48gx	R	ID12200475	Indonesia	2001	22	
48gx	R	ID14500635	Indonesia	2001	45	Black
48g+	R	ID25000299	Indonesia	2002	50	Black

Edited: 17 Mar 2014, 2:00 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

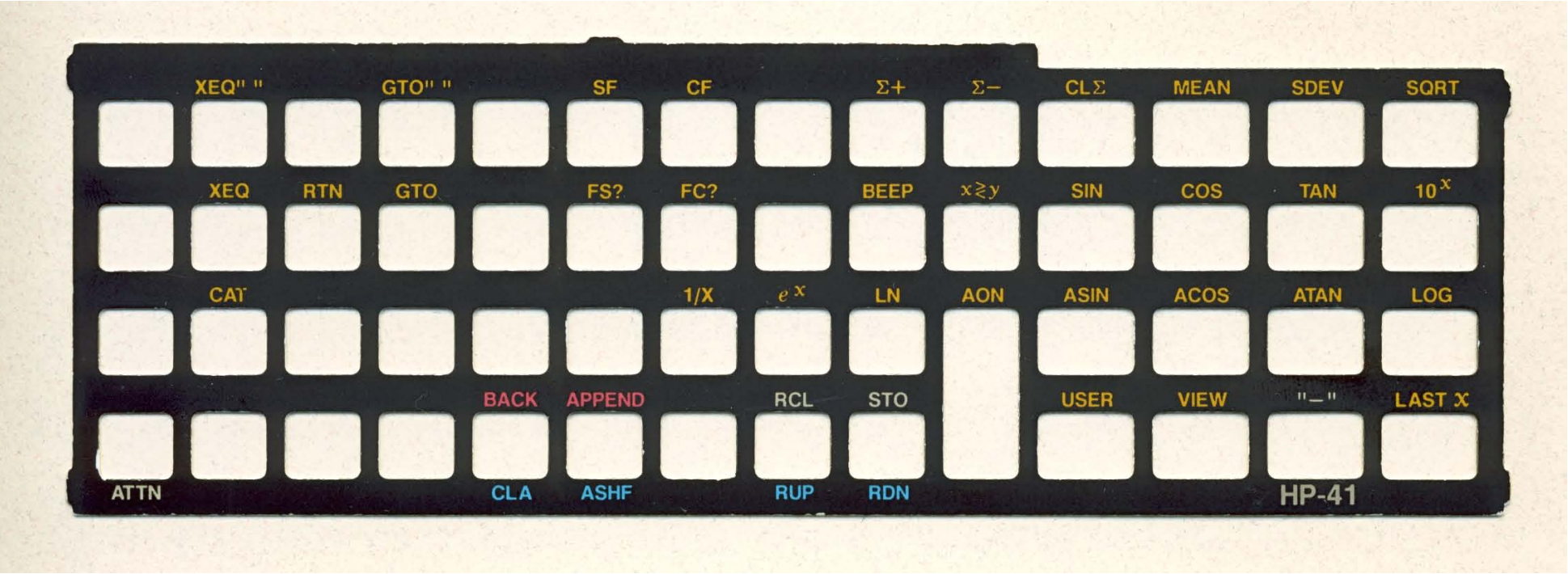
HP Articles Forum

[Return to the Index]
[Previous | Next]

HP 82490A Translator Pac Overlay

Posted by Allen on 8 Feb 2010, 11:32 a.m.

I've gotten a number of requests lately for the 82490A Overlay. Since they are devilishly hard to find, here are two pictures of them, should you wish to make your own.
Images are copyright 2004-2010 Enterhp.com and are not available for commercial use without permission.





Edited: 8 Feb 2010, 11:35 a.m.

Edit Message

Delete Message

Password:

[\[Return to the Message Index \]](#)

GTO

[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

R->P and P->R for the 35s, Reloaded

Posted by [Frido Bohn](#) on 26 Jan 2010, 4:06 a.m.

It is true that the HP35s does not offer a direct access to polar-rectangular conversion. However, you may do this with the capabilities of the calculator by taking advantage of the display of complex numbers as real numbers with an imaginary part (x, iy) or in polar notation ($r, \text{"theta" degrees}$). Be aware of the angular representation: degrees, radians etc accessible via "MODE". Theta is accessible as a blue arrowed key in the second row, 4th column with the appearance of a strikethrough "O".

To convert rectangular to polar coordinates (assuming RPN mode, and angular degrees):

1. Choose "DISPLAY"; item 10 ".0" for the display in polar notation.
2. Enter the rectangular coordinates as xiy ; press "ENTER"

For example type in "8i6", "ENTER" and you will get 10"theta"36.87 instantaneously which represent the polar notation with an r of 10 and an angle of 36.87°.

For the polar to rectangular conversion just do the opposite:

1. Choose "DISPLAY"; item 9 "9" for the display of complex numbers " xiy "
2. Enter the polar coordinates as $r\text{"theta"}^\circ$; press Enter

For example "7.07"theta"45" gives "5i5" (5 on x-axis and 5 on y-axis).

Another way to yield polar coordinates from rectangular notation is to type in " xiy " and calculate "ABS" to obtain r , and "ARG" to obtain the angle.

Note: " xiy " has to be duplicated in the stack as the complex number is destroyed by the calculation of the absolute or argument. After typing "ABS", type " $x<>y$ " then type "ARG". Of course, I should disclose a more elegant approach using the "LAST X"-command.

Addendum:

The conversion of rectangular to polar notation earned only a weak hint in the HP35s User's Guide. On page 9-6 you will find that results of complex numbers are dependent from the option set in "DISPLAY".

If you refer to the training modules for the HP35s ([HP35s Training Modules](#), as per 02-Feb-2010), you will find in the chapter about complex numbers (Part 1) on page 2 a note how to use complex numbers for the conversion, although no example is given.

Interestingly, the apparent lack of a comprehensible rectangular to polar conversion is a main complaint issued by users in the comments on the official site of the HP35s.

Although it has nothing to do with the rectangular to polar conversion, the handling of negative roots by this calculator is whimsical. It might have been quoted elsewhere, but I will repeat this because it fits so well into the context of imaginary numbers.

Ever tried to get a result of the square root of -2 by typing "2" "+/-" "SQRT X"? The answer you get is "SQRT(NEG)", which of course is not false, but is not just what you expected.

Also the use of "2" "+/-" "ENTER" "2" "ROOT X,Y" (accessible via yellow arrow above "SQRT X") is vain. All you get is "INVALID y^x". The 3rd option would be to raise the power of 0.5 by typing "2" "+/-" "ENTER" "0.5" "Y^X". Again the result is "INVALID y^x ". Now try the complex-number notation "2" "+/-" "i" "0" "ENTER" "0.5" "y^x", and miraculously you will get the correct answer "0i1.414". Try the same use of complex notation of a real negative number with "SQRT" or "ROOT X,Y" and you will get "INVALID DATA".

The conclusion is that the HP35s is able to handle the root of negative numbers, but you have to use complex notation and the "y^x"-command. The other commands related to the raise of powers or handling of roots are not useful in this context. For me it appears that the internal circuitry for "SQRT" and "ROOT X,Y" came from the simplest calculator design, whereas the "y^x"-command has a more noble origin. Try "1" "e^x" "PI" "I" "*" "y^x" and you get somewhat like "-1.0i-1.267E-12" which is of course a fair approach of such a calculator to Euler's identity.

Kind regards

Frido

Edited: 2 Feb 2010, 5:47 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Psychrometric calculations on the HP 35s (and 50g)

Posted by [Maarten Ambaum \(Reading, UK\)](#) on 22 Jan 2010, 11:14 a.m.

For those who are interested in meteorology, I wrote a very useable HP 35s program that calculates the relative humidity and vapour density when the temperature and wet bulb temperature are known. I post it here as a nice example of practical use for the HP 35s in science. Other standard methods (computer, tables, charts) are relatively impractical in the field.

It is part of the extra resources that I publish to go with a book on thermal physics of the atmosphere; I will not mention any further details here, lest I get accused of plugging my book. The theory behind wet bulb temperature is somewhat involved and it is therefore impractical to describe it in detail here.

The program description and listing can be found in this [1-page article \(pdf\)](#).

EDIT: I have now added a RPL code to do the same. It is version 1 and does not have the option to work in Fahrenheit (UserRPL apparently does not allow INFORMs with checkboxes or choose boxes, or customized menus), but it does look nicer than the 35s version.

For both versions, see [this webpage](#).

Kind regards,
Maarten Ambaum

Edited: 12 Feb 2010, 10:00 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

R->P & P->R for the 35s

Posted by [Paul Dale](#) on 21 Jan 2010, 5:22 a.m.

The 35s lacks polar to rectangular conversions. These two programs address this short coming.

```
P001 LBL P
P002 FS? 10
P003 GT0 P011
P004: Rv
P005 Rv
P006 eqn REGZ+i*REGT
P007 ARG
P008 LASTx
P009 ABS
P010 RTN
P011: CF 10
P012 XEQ P004
P013 SF 10
P014 RTN
```

```
R001 LBL R
R002 FS? 10
R003 GT0 R012
R004: Rv
R005 Rv
R006 eqn [REGZ*SIN(REGT), REGZ*COS(REGT)]
R007 [1,0]
R008 x<>y
R009 *
R010 eqn LASTx*[0,1]
R011 RTN
R012: CF 10
R013 XEQ R004
R014 SF 10
R015 RTN
```

- Pauli

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

nCr and nPr - a 41-MCODE implementation

Posted by [Ángel Martín](#) on 27 Nov 2009, 10:46 a.m.

You can refer to this forum threads for a more detailed discussion of this topic:

<http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv019.cgi?read=160031#160031>

<http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv013.cgi?read=39655>

Here's the revised code with the following additions:

- Gets the integer part of the input values, forcing them to be positive
- Checks that neither one is Zero, and that $n > r$
- Uses the minimum of $\{r, (n-r)\}$ to expedite the calculation time
- The Out Of Range condition is checked at every multiplication, so it's determined as soon as possible
- The chain of multiplication proceeds right-to-left, with the largest quotients first.
- The algorithm works within the numeric range of the 41. Example: $C(335,167)$ is calculated without problems.
- It doesn't perform any rounding on the results. Partial divisions are done as opposed to calculating first nPr and dividing it by r!

Incidentally, the same checking routines are used by the GCD and LCM functions in the SandMath as well... so there's some nice code reuse.

Hope you enjoy it - pls. send comments and improvements!

Best wishes, ÁM

```

Header  A49D    092    "R"          nCr= nPr / r!
Header  A49E    003    "C"          nCr = PROD{(n-k)/(r-k)}
Header  A49F    00E    "N"          k=0,1,...,(r-2),(r-1)
NCR     A4A0    108    SETF 8        Angel Martin
NCR     A4A1    379    PORT DEP:     Naturalize inputs
NCR     A4A2    03C    XQ           Integer & Positive
NCR     A4A3    0F4    ->A4F4 [NATXY]
NCR     A4A4    070    N=C ALL      r
NCR     A4A5    0B8    READ 2(Y)    n
NCR     A4A6    10E    A=C ALL
NCR     A4A7    085    ?NC XQ       Calculates (n-r)
NCR     A4A8    050    ->1421      [Y_MINUS_X]
NCR     A4A9    2EE    ?C#0 ALL     is n=r?
NCR     A4AA    053    JNC +10d     case r=n
NCR     A4AB    128    WRIT 4(L)    see if n>r
NCR     A4AC    23E    C=C+1 MS     Carry set if n<r
NCR     A4AD    30D    ?C GO
NCR     A4AE    063    ->18C3 [ERR0]
NCR     A4AF    0F8    READ 3(X)
NCR     A4B0    10E    A=C ALL      r
NCR     A4B1    085    ?NC XQ       Calculates r-(n-r)
NCR     A4B2    050    ->1421 [Y_MINUS_X]
NCR     A4B3    23E    C=C+1 MS     r-(n-r) = 2r-n
NCR     A4B4    05F    JC +11d      Carry set if (2r-n)<0, or r<(n-r)

```

NCR	A4B5	138	READ 4(L)	
NCR	A4B6	0E8	WRIT 3(X)	
NCR	A4B7	043	JNC +08	
Header	A4B8	092	"R"	nPr = PROD{n-k}
Header	A4B9	010	"P"	k=0,1,...,(r-2),(r-1)
Header	A4BA	00E	"N"	Angel Martin
NPR	A4BB	104	CLRF 8	
NPR	A4BC	379	PORT DEP:	Naturalize inputs
NPR	A4BD	03C	XQ	Integer & Positive
NPR	A4BE	0F4	->A4F4 [NATXY]	
NPR	A4BF	04E	C=0 ALL	
NPR	A4C0	35C	PT= 12	builds "1" in C
NPR	A4C1	050	LD@PT- 1	
NPR	A4C2	268	WRIT 9(Q)	initial product
NPR	A4C3	0F8	READ 3(X)	min[r,(n-r)]
NPR	A4C4	10E	A=C ALL	r
NPR	A4C5	04E	C=0 ALL	
NPR	A4C6	2DC	PT= 13	Build "-1" in C
NPR	A4C7	250	LD@PT- 9	-
NPR	A4C8	050	LD@PT- 1	1
NPR	A4C9	01D	?NC XQ	Adds normalized A and C
NPR	A4CA	060	->1807 [AD2_10]	
NPR	A4CB	070	N=C ALL	(r-k), index counter
NPR	A4CC	2BE	C=-C-1 MS	-(r-k)
NPR	A4CD	3C4	ST=0	Clears Carry if set
NPR	A4CE	128	WRIT 4(L)	-(r-k)
NPR	A4CF	10E	A=C ALL	-(r-k)
NPR	A4D0	0B8	READ 2(Y)	n
NPR	A4D1	01D	?NC XQ	Adds normalized A and C
NPR	A4D2	060	->1807 [AD2_10]	
NPR	A4D3	10C	?FSET 8	
NPR	A4D4	063	JNC +12d	skip if nPr case
NPR	A4D5	228	WRIT 8(P)	save [n-(r-k)] it in P
NPR	A4D6	0F8	READ 3(X)	r
NPR	A4D7	10E	A=C ALL	
NPR	A4D8	138	READ 4(L)	-(r-k)
NPR	A4D9	01D	?NC XQ	Adds normalized A and C
NPR	A4DA	060	->1807 [AD2_10]	
NPR	A4DB	10E	A=C ALL	r-(r-k)=k
NPR	A4DC	238	READ 8(P)	:-{
NPR	A4DD	0AE	A<>C ALL	
NPR	A4DE	261	?NC XQ	Divides A over C
NPR	A4DF	060	->1898 [DV2_10]	
NPR	A4E0	10E	A=C ALL	n-[(r-k)-1]
NPR	A4E1	278	READ 9(Q)	partial product PP
NPR	A4E2	135	?NC XQ	it *uses* M !!
NPR	A4E3	060	->184D [MP2_10]	
NPR	A4E4	2F6	?C#0 XS	See if we overflow?
NPR	A4E5	289	?C GO	"Out of Range"
NPR	A4E6	003	->00A2 [ERROF]	
NPR	A4E7	268	WRIT 9(Q)	PP*{n-[(r-k)-1]}
NPR	A4E8	0B0	C=N ALL	(r-k)-1
NPR	A4E9	2EE	?C#0 ALL	
NPR	A4EA	2D7	JC -38d	loop back
NPR	A4EB	278	READ 9(Q)	
NPR	A4EC	0EE	B<>C ALL	final solution
NPR	A4ED	391	?NC GO	(REQUIRES Value in B)
NPR	A4EE	002	->00E4 [DROPST]	
Header	A4EF	099	"Y"	
Header	A4F0	018	"X"	
Header	A4F1	014	"T"	Naturalize inputs
Header	A4F2	001	"A"	INT, ABS, #0
Header	A4F3	00E	"N"	
NATXY	A4F4	0F8	READ 3(X)	r
NATXY	A4F5	0B8	READ 2(Y)	n
NATXY	A4F6	351	?NC XQ	(includes SETDEC)
NATXY	A4F7	050	->14D4 [CHK_NO_S1]	
NATXY	A4F8	088	SETF 5	Integer
NATXY	A4F9	0ED	?NC XQ	Take Integer part
NATXY	A4FA	064	->193B [INTFRC]	
NATXY	A4FB	05E	C=0 MS	make it positive >0
NATXY	A4FC	2EE	?C#0 ALL	
NATXY	A4FD	03B	JNC +07	
NATXY	A4FE	0A8	WRIT 2(Y)	n (integer, >0)
NATXY	A4FF	0F8	READ 3(X)	r
NATXY	A500	0ED	?NC XQ	Take Integer part
NATXY	A501	064	->193B [INTFRC]	
NATXY	A502	05E	C=0 MS	make it positive >0
NATXY	A503	2EE	?C#0 ALL	Carry set if NOT Zero
NATXY	A504	30D	?NC GO	

```
NATXY  A505  062  ->18C3 [ERR0]
NATXY  A506  0E8  WRIT 3(X)   r (integer, >0)
NATXY  A507  3E0  RTN
```

Edited: 25 Oct 2010, 1:28 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Fast DiGamma Function for Positive and Negative Real Arguments

Posted by [Steven Thomas Smith](#) on 26 Sept 2009, 6:06 a.m.

This code computes the logarithmic derivative of the Gamma function, $\text{digamma}(x) = \Gamma'(x)/\Gamma(x)$. It works for all real arguments, positive and negative, and is robust for very small arguments. It simply uses the reflection formula

$$\text{digamma}(x+1) = \text{digamma}(x) + 1/x$$

to move to a region where the [asymptotic expansion](#) for digamma yields full precision. When x is negative, it uses the analytic continuation term. It is based on [Jean-Marc Baillard's PSI](#), which uses the asymptotic expansion for positive arguments.

```

01 LBL "DIGAMM"
02 CLA
03 8
04 X<>Y
05 X>0?
06 GTO 01
07 STO M
08 RAD
09 X<>Y
10 CHS
11 X<>Y
12 PI
13 *
14 TAN
15 1/X
16 PI
17 *
18 X<> M
19 LBL 00
20 E
21 -
22 1/X
23 ST- M
24 X<> L
25 X>Y?
26 GTO 00
27 GTO 02
28 LBL 01
29 1/X
30 ST+ M
31 X<> L
32 E
33 +
34 X<Y?
35 GTO 01
36 LBL 02
37 1/X
38 STO N
39 X^2
40 ENTER
41 STO Z
42 20

```



```
43 /
44 21
45 1/X
46 -
47 *
48 E -1
49 +
50 *
51 E
52 -
53 *
54 12
55 /
56 RCL N
57 ABS
58 LN
59 RCL N
60 2
61 /
62 +
63 -
64 RCL M
65 -
66 END
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Fast Incomplete Gamma/Exponential Integral/Error Functions With Lentz's Method

Posted by [Steven Thomas Smith](#) on 23 Sept 2009, 1:28 p.m.

The incomplete gamma function $\gamma(a, x) = \int_0^x e^{-t} t^{a-1} dt$, error function $\operatorname{erf}(x) = (2/\sqrt{\pi}) \int_0^x e^{-t^2} dt$ and exponential integrals $E_n(x) = x^{n-1} \int_x^\infty e^{-t} t^{-n} dt$ are all closely related. The exponential integral $E_1(x)$ is also important because it provides the exponential integral $\operatorname{Ei}(x) = \int_{-\infty}^x e^{-t} t^{-1} dt = -\operatorname{Re}(E_1(-x))$ (ignoring the imaginary $-i\pi$ from integrating halfway around the pole at the origin) and the logarithmic integral $\operatorname{li}(x) = \operatorname{Ei}(\log x)$. The error and incomplete gamma functions $P(a, x) = \gamma(a, x)/\Gamma(a)$ [lower tail] and $Q(a, x) = 1 - P(a, x)$ [upper tail, "regularized" by dividing by $\Gamma(a)$], are commonly encountered in the cumulative probability distribution of the normal (Gaussian) density and the chi-squared density functions, respectively. $E_n(x)$ is essentially $\Gamma(-n+1, x)$ with the singularities at the nonpositive integers removed.

The continued fraction expansions of all these functions can be derived by integrating by parts the integral $f(a, b) = (1/\Gamma(a)) \int_0^\infty e^{-t} t^{a-1} (1+xt)^{-b} dt$ (see Wall, [Analytic Theory of Continued Fractions](#), p. 354ff), which results in the following similar continued fractions:

$$\Gamma(a, x) = \Gamma(a) - \gamma(a, x) = e^{-x} x^a \cfrac{1/x}{1 +} \cfrac{(1-a)/x}{1 +} \cfrac{1/x}{1 +} \cfrac{(2-a)/x}{1 +} \cfrac{2/x}{1 +} \dots$$

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = \cfrac{e^{-x^2}}{\sqrt{\pi}} \cfrac{1}{x +} \cfrac{1/2}{x +} \cfrac{1}{x +} \cfrac{3/2}{x +} \cfrac{2}{x +} \dots$$

$$E_n(x) = e^{-x} \cfrac{1/x}{1 +} \cfrac{n/x}{1 +} \cfrac{1/x}{1 +} \cfrac{(n+1)/x}{1 +} \cfrac{2/x}{1 +} \dots$$

The simple, similar form of these continued fractions provides a fast, efficient algorithm for computing these functions for x larger than 2 or so. This is especially important for the exponential integrals, whose series representation alternates for positive x and therefore causes cancellation errors for moderately sized x .

Though backward recursion through the continued fraction is the most efficient, it is not possible to pre-compute the number of terms required to achieve the desired precision with two-parameter functions such as $\gamma(a, x)$ and $E_n(x)$. Therefore, forward recursion is used, the best option being [Lentz's Method](#) (see [Numerical Recipes](#), 2d ed., p. 207ff). Briefly, it is easy to show that the numerator A_n and denominator B_n of the continued fraction

$$b_0 + \cfrac{a_1}{b_1 +} \cfrac{a_2}{b_2 +} \dots$$

satisfy the same second-order linear recurrence relation:

$$A_n = b_n A_{n-1} + a_n A_{n-2}; A_0 = b_0, A_{-1} = 1$$

$$B_n = b_n B_{n-1} + a_n B_{n-2}; B_0 = 1, B_{-1} = 0$$

Rather than computing A_n , B_n , and the n th order approximate A_n/B_n directly, the ratios $C_n = A_n/A_{n-1}$, $D_n = B_n/B_{n-1}$ are used, which satisfy the simpler first-order nonlinear recursion:

$$C_n = b_n + a_n/C_{n-1}; C_0 = \text{HUGE\_VAL}$$

$$D_n = b_n + a_n/D_{n-1}; D_0 = 1$$

The n th approximate is then given by repeated multiplication $(A_n/B_n) = (C_n/D_n)(A_{n-1}/B_{n-1})$. Convergence is achieved when $C_n/D_n = 1$ to machine precision, or $C_n = D_n$ to machine precision. The value for $C_0 = \text{HUGE\_VAL}$ need simply be large enough so that $C_1 = b_1$ to machine precision, which for these functions at moderately sized x and larger means that $\text{HUGE\_VAL} = \text{E99}$ on the HP41 is sufficient. In general, the modified Lentz's algorithm requires checking that neither C_n nor D_n becomes too small and adjusting them to control for underflow if they do, but these continued fractions are sufficiently well-behaved and this robustness checking is not required here.

The following HP41 algorithms compute all these functions using either a series (small-ish x), a backward recursion continued fraction if the number of necessary terms is known, or a forward recursion continued fraction using Lentz's Method, all in status registers XYZT/LMNO. Some of the series code is borrowed or modified from other sources, such as [Jean-Marc Baillard's excellent collection of functions](#). It requires a GAMMA function (such as [this one](#)), and a digamma function for the higher order exponential integrals (such as [Jean-Marc Baillard's PSI](#)). Actual numerical software would use Chebyshev and/or rational approximations over many ranges of arguments -- but these practical approximation methods are much less interesting to understand and code!

The incomplete gamma function GAMMINC computes both upper and lower tails without underflow errors for small or large arguments, and for both ordinary and regularized functions.

a ENTER x XEQ GAMMINC produces the stacked results

T: Gamma(a, x)
 Z: gamma(a, x)
 Y: $Q(a, x)$
 X: $P(a, x)$

Due to the efficient continued fraction algorithm performed in the status registers, GAMMINC runs about twice as fast as [IGF](#).

The error function computes both upper and lower tails without underflow errors for small or large arguments. x XEQ ERF produces the stacked results

Z: $\text{erfcx}(x) = \exp(x^2) \cdot \text{erfc}(x)$ for larger x
 Y: $\text{erfc}(x)$
 X: $\text{erf}(x)$

The exponential integrals $E_1(x)$ and $E_n(x)$ are handled separately so that the special case $\text{Ei}(x) = -\text{Re}(E_1(-x))$ with negative arguments can be addressed. The continued fraction for $E_1(x)$ is not convergent for negative x ; therefore, an asymptotic expansion is used for large negative x , and both Ei and li may then be computed with the simple subroutines:

```
01 LBL "EI"
02 CHS
03 XEQ "E1"
04 CHS
05 END
```

```
01 LBL "LI"
02 LN
03 XEQ "EI"
04 END
```

The HP41 code for $\Gamma(a,x)$, $\gamma(a,x)$, $Q(a,x)$, and $P(a,x)$ [a ENTER x XEQ GAMMINC]:

```
01 LBL "GAMMINC"
02 STO 00
03 X<>Y
04 STO 01
05 XEQ "GAMMA"
06 SF 05
07 CF 06
08 X<> 01
09 X<0?
10 SF 06
11 STO N
12 RCL 00
13 STO M
14 X<=Y?
15 CF 05
16 5
17 FS?C 06
18 GTO 00
19 X>Y?
20 CF 05
21 LBL 00
22 1/X
23 X>Y?
24 CF 05
25 LBL 00
26 RDN
27 X!=0?
28 GTO 00
29 RCL 01
30 .
31 E
32 .
33 RTN
34 LBL 00
35 RCL N
36 RCL M
37 LN
38 *
39 RCL M
40 -
41 E^X
42 FS? 05
43 GTO 02
44 RCL N
45 /
46 STO 0
47 LBL 01
48 RCL M
49 RCL N
50 E
51 +
52 STO N
53 /
54 RCL 0
55 *
56 STO 0
57 +
58 X!=Y?
59 GTO 01
60 GTO 07
61 LBL 02
62 E
63 X<> N
64 STO L
65 X<>Y
66 E99
67 X<> M
68 RCL N
69 RCL Z
70 RCL Z
71 ST/ Y
72 RDN
```

```
73 ST0 0
74 CLX
75 RCL M
76 RCL N
77 LBL 03
78 X=Y?
79 GT0 06
80 R^
81 R^
82 RCL X
83 LASTX
84 -
85 R^
86 ST/ Y
87 CLX
88 RCL Y
89 SF 06
90 LBL 04
91 X<> M
92 ST/ M
93 X<>Y
94 X<> N
95 ST/ N
96 CLX
97 E
98 ST+ M
99 ST+ N
100 R^
101 R^
102 RCL M
103 ST* 0
104 RCL N
105 ST/ 0
106 X=Y?
107 GT0 06
108 FC?C 06
109 GT0 05
110 R^
111 R^
112 RCL X
113 RCL Z
114 ST/ Y
115 CLX
116 RCL Y
117 GT0 04
118 LBL 05
119 ISG Z
120 "" ;F0
121 GT0 03
122 LBL 06
123 R^
124 ST0 M
125 LASTX
126 ST0 N
127 RCL 0
128 LBL 07
129 RCL 01
130 X<>Y
131 -
132 LASTX
133 FS?C 05
134 X<>Y
135 ST0 0
136 RCL Y
137 RCL 01
138 ST/ Y
139 ST/ 0
140 CLX
141 RCL 0
142 END
```

The HP41 code for erf(x) [x XEQ ERF]:

```
01 LBL "ERF"
02 CF 05
03 CF 06
04 X<=0?
05 SF 06
06 ABS
```

```

07 STO M
08 6
09 LN
10 X<=Y?
11 GTO 03
12 RCL M
13 X^2
14 STO N
15 E
16 STO 0
17 .
18 LASTX
19 .
20 LBL 01
21 X=Y?
22 GTO 02
23 ISG Z
24 ""
25 CLX
26 RCL M
27 RCL N
28 *
29 RCL Z
30 /
31 RCL 0
32 *
33 2
34 ST+ 0
35 CLX
36 RCL 0
37 /
38 CHS
39 STO M
40 X<>Y
41 ST+ Y
42 GTO 01
43 LBL 02
44 ST+ X
45 PI
46 SQRT
47 /
48 FS?C 06
49 CHS
50 GTO 09
51 LBL 03
52 SF 05
53 CLX
54 4
55 X<=Y?
56 GTO 05
57 24
58 .
59 LBL 04
60 RCL M
61 +
62 RCL Y
63 X<>Y
64 ST+ X
65 /
66 DSE Y
67 GTO 04
68 RCL M
69 +
70 1/X
71 GTO 08
72 LBL 05
73 E99
74 X<> M
75 STO N
76 ENTER
77 1/X
78 STO 0
79 CLX
80 SIGN
81 RCL M
82 RCL Y
83 LBL 06
84 X=Y?
85 GTO 07
86 R^
87 R^

```

```

88 ENTER
89 ENTER
90 2
91 /
92 ENTER
93 X<> M
94 ST/ M
95 X<>Y
96 X<> N
97 ST/ N
98 R^
99 ST+ M
100 ST+ N
101 R^
102 RCL M
103 RCL N
104 /
105 ST* 0
106 E
107 ST+ Z
108 GTO 06
109 LBL 07
110 R^
111 STO M
112 RCL 0
113 LBL 08
114 PI
115 SQRT
116 /
117 STO Y
118 RCL M
119 X^2
120 CHS
121 E^X
122 *
123 FC?C 06
124 GTO 09
125 2
126 -
127 CHS
128 LBL 09
129 ENTER
130 ENTER
131 E
132 -
133 CHS
134 FC?C 05
135 X<>Y
136 END

```

The function E1 also uses the synthetic text line "\05\77\21\56\64\99\99" (bytes F7 05 77 21 56 64 99 99), RCL M, which stores Euler's constant 0.5772156649 in the X register. The HP41 code for $E_1(x)$ [x XEQ E1]:

```

01 LBL "E1"
02 STO M
03 5
04 LN
05 X<=Y?
06 GTO 03
07 CLX
08 32
09 CHS
10 X>Y?
11 GTO 08
12 RCL M
13 CHS
14 STO M
15 STO 0
16 E
17 STO N
18 .
19 RCL 0
20 LBL 01
21 X=Y?
22 GTO 02
23 RCL M
24 RCL N
25 *
26 ISG N

```



```

27 "" ;F0
28 RCL N
29 X^2
30 /
31 ST* 0
32 RDN
33 RCL X
34 RCL 0
35 +
36 GT0 01
37 LBL 02
38 CHS
39 RCL M
40 ABS
41 LN
42 -
43 " w!Vd " ;"\05\77\21\56\64\99\99"
44 RCL M
45 -
46 RTN
47 LBL 03
48 CLX
49 3
50 X<=Y?
51 GT0 06
52 27
53 E
54 LBL 04
55 SF 06
56 LBL 05
57 RCL M
58 *
59 RCL Y
60 X<>Y
61 /
62 E
63 +
64 FS?C 06
65 GT0 05
66 DSE Y
67 GT0 04
68 RCL M
69 E^X
70 LASTX
71 *
72 *
73 1/X
74 RTN
75 LBL 06
76 RCL M
77 CHS
78 E^X
79 RCL M
80 /
81 ST0 0
82 E99
83 X<> M
84 E
85 ENTER
86 ST0 N
87 RCL M
88 LBL 07
89 X=Y?
90 GT0 10
91 R^
92 R^
93 ENTER
94 ENTER
95 RCL T
96 /
97 ST0 L
98 RDN
99 SF 06
100 LBL 08
101 LASTX
102 ENTER
103 X<> M
104 ST/ M
105 X<>Y
106 X<> N
107 ST/ N

```

```
108 CLX
109 E
110 ST+ M
111 ST+ N
112 R^
113 R^
114 RCL M
115 LASTX
116 X<> N
117 /
118 ST* 0
119 LASTX
120 X<> N
121 STO L
122 CLX
123 E
124 X=Y?
125 GTO 10
126 FC?C 06
127 GTO 09
128 R^
129 R^
130 GTO 08
131 LBL 09
132 ISG Z
133 "" ;F0
134 GTO 07
135 LBL 10
136 RCL 0
137 RTN
138 LBL 11
139 13
140 STO N
141 RCL M
142 CHS
143 STO M
144 1/X
145 STO 0
146 E
147 GTO 13
148 LBL 12
149 RCL N
150 *
151 RCL 0
152 *
153 E
154 +
155 LBL 13
156 DSE N
157 GTO 12
158 LBL 13
159 RCL M
160 E^X
161 *
162 RCL M
163 /
164 CHS
165 END
```

The HP41 code for $E_n(x)$ [n ENTER x XEQ EN]:

```
01 LBL "EN"
02 X<>Y
03 X!=0?
04 GTO 00
05 X<>Y
06 E^X
07 LASTX
08 *
09 1/X
10 RTN
11 LBL 00
12 E
13 X!=Y?
14 GTO 00
15 RCL Z
16 XEQ "E1"
17 RTN
18 LBL 00
```

```

19 RDN
20 X<>Y
21 5
22 LN
23 X<=Y?
24 GTO 04
25 RDN
26 STO 00
27 X<>Y
28 STO 01
29 XEQ "PSI"
30 RCL 00
31 STO M
32 LN
33 -
34 RCL M
35 CHS
36 RCL 01
37 E
38 STO 0
39 -
40 CHS
41 STO N
42 CHS
43 Y^X
44 LASTX
45 FACT
46 /
47 *
48 STO 01
49 .
50 RCL N
51 1/X
52 RCL Y
53 LBL 01
54 X=Y?
55 GTO 03
56 CLX
57 SIGN
58 ST+ Z
59 ST+ N
60 CLX
61 RCL N
62 X!=0?
63 GTO 02
64 CLX
65 RCL M
66 CHS
67 RCL Z
68 /
69 ST* 0
70 CLX
71 GTO 01
72 LBL 02
73 CLX
74 RCL M
75 CHS
76 RCL Z
77 /
78 RCL N
79 /
80 ST* 0
81 CLX
82 RCL Y
83 RCL 0
84 +
85 X<>Y
86 RCL N
87 ST* 0
88 RDN
89 GTO 01
90 LBL 03
91 CHS
92 RCL 01
93 +
94 RTN
95 LBL 04
96 RDN
97 CHS
98 E^X
99 LASTX

```

```
100 CHS
101 /
102 STO 0
103 CLX
104 LASTX
105 X<>Y
106 STO L
107 CLX
108 E
109 E99
110 STO M
111 RCL Y
112 STO N
113 LBL 05
114 X=Y?
115 GTO 08
116 R^
117 R^
118 X<> L
119 ENTER
120 ENTER
121 RCL T
122 ST/ Y
123 RDN
124 ENTER
125 SF 06
126 LBL 06
127 X<> M
128 ST/ M
129 X<>Y
130 X<> N
131 ST/ N
132 CLX
133 E
134 ST+ M
135 ST+ N
136 R^
137 R^
138 RCL M
139 ST* 0
140 RCL N
141 ST/ 0
142 X=Y?
143 GTO 08
144 FC?C 06
145 GTO 07
146 R^
147 R^
148 X<> L
149 ENTER
150 ENTER
151 RCL T
152 ST/ Y
153 RDN
154 ENTER
155 GTO 06
156 LBL 07
157 ISG Z
158 "" ;F0
159 ISG L
160 "" ;F0
161 GTO 05
162 LBL 08
163 RCL 0
164 END
```

Edited: 23 Sept 2009, 3:03 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

EIGEN: Characteristic equation+eigenvalues for order 2,3,4,5

Posted by [Steven Thomas Smith](#) on 1 Sept 2009, 12:39 p.m.

EIGEN computes the characteristic equation and eigenvalues of matrices of order 2, 3, 4, and 5. It computes the coefficients of the characteristic polynomial directly, then solves for the roots of the polynomial. Routines from the MATHSTAT (or MATH I) module are used -- EIGEN expects that MATRIX is run first to enter the matrix elements, then both DET (XROM 01,06) and ROOTS (XROM 01,12) are called. It would be straightforward to edit the register numbers to use the (more efficient) matrix storage routines from the Advantage PAC.

Compared to alternate approaches such as Samuelson's formula (see, e.g., [Jean-Marc Baillard's program CP](#)), computing the polynomial coefficients directly for low-order matrices has the advantage of speed at the cost of longer code, at least for order 5 matrices, and limited matrix order. Note that the method used by EIGEN is in general the *opposite* approach of an algorithm that obtains the best numerical accuracy (where the eigenvalues are computed first, then the polynomial coefficients; see Matlab's POLY function). However, the accuracy issues associated with computing polynomial coefficients and the ill-conditioning of computing polynomial roots are not a concern for very low order matrices -- compare Matlab's ROOTS(POLY(1:5)) and ROOTS(POLY(1:21)). (And computations for higher order matrices are best done on a computer, if only for data entry!)

To load EIGEN, first install the MATHSTAT (or MATH I) module and XEQ SIZE 090 to allocate sufficient program and memory registers. To run EIGEN, XEQ MATRIX (MATHSTAT/MATH I module) and enter the matrix elements as prompted. Then XEQ EIGEN. The coefficients of the characteristic polynomial will first be displayed, then their roots, the eigenvalues. R/S through each of the displays.

I was curious how this code compared to SANDMATH'-5 packages's compiled M-code "CHRPOL", which I believe has the identical functionality. Using the Wilkinson test matrix,

```
>> W5 = wilkinson(5)
```

```
W5 =
    2    1    0    0    0
    1    1    1    0    0
    0    1    0    1    0
    0    0    1    1    1
    0    0    0    1    2
```

which has characteristic polynomial and eigenvalues (double precision),

```
>> poly(W5), eig(W5).'
```

```
ans =
 1.0000000000000000 -6.0000000000000000 8.9999999999999996 4.0000000000000008 -
13.0000000000000007 4.0000000000000000
ans =
-1.114907541476756 0.381966011250105 1.254101688365053 2.618033988749895
2.860805853111704
```

This code return the eigenvalues

```
-1.114907542 0.3819660170 1.254101674 2.618034045 2.860805070
```

(maximum error 8e-7) after about 37 s to compute the characteristic polynomial, and 115 s for MATH:ROOTS to compute its roots (using the i41CX's ridiculously slow "Default Speed").

SANDMATH-5:CHRPOL returns the correct characteristic polynomial after 7+ minutes. However, CHRPOL doesn't return the correct eigenvalues if you hit R/S after the characteristic polynomial coefficients are shown. Perhaps I have not called CHRPOL correctly. A call to SANDMATH-5:POLYN could be used to compute eigenvalues (after reentering the coefficients), but I stopped waiting to verify results after another 9+ more minutes and cranking up the i41CX's speed to the max.

I believe that the low order cases ($N = 2, 3, 4, 5$) may be sped up considerably by using specialized code, as done here.

The code below is generated by Antonio Lagana's i41CX+ iPhone emulator.

```

01 LBL "EIGEN"
02 RCL 14
03 6
04 X<=Y?
05 GTO 01
06 X<>Y
07 STO 59
08 GTO IND X
09 LBL 02
10 LBL 20
11 RCL 15
12 RCL 18
13 *
14 RCL 16
15 RCL 17
16 *
17 -
18 STO 00
19 LBL 21
20 RCL 15
21 RCL 18
22 +
23 CHS
24 STO 01
25 1
26 STO 02
27 GTO 06
28 LBL 03
29 LBL 31
30 RCL 19
31 RCL 23
32 +
33 RCL 15
34 *
35 RCL 19
36 RCL 23
37 *
38 +
39 RCL 16
40 RCL 18
41 *
42 -
43 RCL 17
44 RCL 21
45 *
46 -
47 RCL 20
48 RCL 22
49 *
50 -
51 STO 31
52 LBL 32
53 RCL 15
54 RCL 19
55 +
56 RCL 23
57 +
58 CHS
59 STO 32
60 LBL 30
61 SF 02
62 XROM 01,06

```


63 CHS
64 STO 00
65 RCL 31
66 STO 01
67 RCL 32
68 STO 02
69 1
70 STO 03
71 GT0 06
72 LBL 04
73 LBL 41
74 RCL 25
75 RCL 30
76 *
77 RCL 26
78 RCL 29
79 *
80 -
81 RCL 20
82 *
83 RCL 26
84 RCL 28
85 *
86 RCL 24
87 RCL 30
88 *
89 -
90 RCL 21
91 *
92 +
93 RCL 24
94 RCL 29
95 *
96 RCL 25
97 RCL 28
98 *
99 -
100 RCL 22
101 *
102 +
103 STO 32
104 RCL 25
105 RCL 30
106 +
107 RCL 20
108 *
109 RCL 25
110 RCL 30
111 *
112 +
113 RCL 21
114 RCL 24
115 *
116 -
117 RCL 22
118 RCL 28
119 *
120 -
121 RCL 26
122 RCL 29
123 *
124 -
125 STO 33
126 RCL 15
127 *
128 ST+ 32
129 RCL 25
130 RCL 30
131 +
132 RCL 19
133 *
134 RCL 21
135 RCL 23
136 *
137 -
138 RCL 22
139 RCL 27
140 *
141 -
142 RCL 16
143 *

144	ST -	32
145	RCL	20
146	RCL	30
147	+	
148	RCL	23
149	*	
150	RCL	19
151	RCL	24
152	*	
153	-	
154	RCL	26
155	RCL	27
156	*	
157	-	
158	RCL	17
159	*	
160	ST -	32
161	RCL	20
162	RCL	25
163	+	
164	RCL	27
165	*	
166	RCL	19
167	RCL	28
168	*	
169	-	
170	RCL	23
171	RCL	29
172	*	
173	-	
174	RCL	18
175	*	
176	ST -	32
177	RCL	32
178	CHS	
179	ST0	41
180	LBL	42
181	RCL	20
182	RCL	25
183	+	
184	RCL	30
185	+	
186	RCL	15
187	*	
188	RCL	33
189	+	
190	RCL	16
191	RCL	19
192	*	
193	-	
194	RCL	17
195	RCL	23
196	*	
197	-	
198	RCL	18
199	RCL	27
200	*	
201	-	
202	ST0	42
203	LBL	43
204	RCL	15
205	RCL	20
206	+	
207	RCL	25
208	+	
209	RCL	30
210	+	
211	CHS	
212	ST0	43
213	LBL	40
214	SF 02	
215	XR0M	01, 06
216	ST0	00
217	RCL	41
218	ST0	01
219	RCL	42
220	ST0	02
221	RCL	43
222	ST0	03
223	1	
224	ST0	04

225	GTO	06
226	LBL	05
227	LBL	51
228	RCL	33
229	RCL	39
230	+	
231	STO	78
232	RCL	27
233	*	
234	RCL	33
235	RCL	39
236	*	
237	+	
238	RCL	28
239	RCL	32
240	*	
241	-	
242	RCL	29
243	RCL	37
244	*	
245	-	
246	RCL	34
247	RCL	38
248	*	
249	-	
250	STO	60
251	RCL	21
252	*	
253	RCL	78
254	RCL	26
255	*	
256	RCL	28
257	RCL	31
258	*	
259	-	
260	RCL	29
261	RCL	36
262	*	
263	-	
264	STO	62
265	RCL	22
266	*	
267	-	
268	RCL	27
269	RCL	39
270	+	
271	STO	79
272	RCL	31
273	*	
274	RCL	26
275	RCL	32
276	*	
277	-	
278	RCL	34
279	RCL	36
280	*	
281	-	
282	STO	63
283	RCL	23
284	*	
285	-	
286	RCL	27
287	RCL	33
288	+	
289	STO	80
290	RCL	36
291	*	
292	RCL	26
293	RCL	37
294	*	
295	-	
296	RCL	31
297	RCL	38
298	*	
299	-	
300	STO	64
301	RCL	24
302	*	
303	-	
304	STO	84
305	RCL	33

306 RCL 39
307 *
308 RCL 34
309 RCL 38
310 *
311 -
312 STO 72
313 RCL 27
314 *
315 RCL 32
316 RCL 39
317 *
318 RCL 34
319 RCL 37
320 *
321 -
322 STO 73
323 RCL 28
324 *
325 -
326 RCL 32
327 RCL 38
328 *
329 RCL 33
330 RCL 37
331 *
332 -
333 STO 74
334 RCL 29
335 *
336 +
337 STO 61
338 RCL 84
339 +
340 RCL 15
341 *
342 STO 51
343 RCL 60
344 RCL 20
345 *
346 RCL 28
347 RCL 30
348 *
349 RCL 29
350 RCL 35
351 *
352 +
353 STO 69
354 CHS
355 RCL 78
356 RCL 25
357 *
358 +
359 STO 66
360 RCL 22
361 *
362 -
363 RCL 25
364 RCL 32
365 *
366 RCL 34
367 RCL 35
368 *
369 +
370 STO 70
371 CHS
372 RCL 79
373 RCL 30
374 *
375 +
376 STO 67
377 RCL 23
378 *
379 -
380 RCL 25
381 RCL 37
382 *
383 RCL 30
384 RCL 38
385 *
386 +

387 ST0 71
388 CHS
389 RCL 80
390 RCL 35
391 *
392 +
393 ST0 68
394 RCL 24
395 *
396 -
397 RCL 16
398 *
399 ST- 51
400 RCL 62
401 RCL 20
402 *
403 RCL 66
404 RCL 21
405 *
406 -
407 RCL 23
408 RCL 31
409 *
410 RCL 24
411 RCL 36
412 *
413 +
414 RCL 34
415 RCL 38
416 *
417 +
418 RCL 33
419 RCL 39
420 *
421 -
422 RCL 25
423 *
424 +
425 RCL 23
426 RCL 26
427 *
428 RCL 29
429 RCL 38
430 *
431 +
432 RCL 28
433 RCL 39
434 *
435 -
436 RCL 30
437 *
438 -
439 RCL 24
440 RCL 26
441 *
442 RCL 28
443 RCL 34
444 *
445 +
446 RCL 29
447 RCL 33
448 *
449 -
450 RCL 35
451 *
452 -
453 RCL 17
454 *
455 ST+ 51
456 RCL 63
457 RCL 20
458 *
459 RCL 67
460 RCL 21
461 *
462 -
463 RCL 22
464 RCL 26
465 *
466 RCL 24
467 RCL 36

468 *
469 +
470 RCL 29
471 RCL 37
472 *
473 +
474 RCL 27
475 RCL 39
476 *
477 -
478 RCL 30
479 *
480 +
481 RCL 22
482 RCL 31
483 *
484 RCL 34
485 RCL 37
486 *
487 +
488 RCL 32
489 RCL 39
490 *
491 -
492 RCL 25
493 *
494 -
495 RCL 24
496 RCL 31
497 *
498 RCL 29
499 RCL 32
500 *
501 +
502 RCL 27
503 RCL 34
504 *
505 -
506 RCL 35
507 *
508 -
509 RCL 18
510 *
511 ST+ 51
512 RCL 64
513 RCL 20
514 *
515 RCL 68
516 RCL 21
517 *
518 -
519 RCL 22
520 RCL 26
521 *
522 RCL 23
523 RCL 31
524 *
525 +
526 ST0 81
527 RCL 28
528 RCL 32
529 *
530 +
531 RCL 27
532 RCL 33
533 *
534 -
535 RCL 35
536 *
537 +
538 RCL 22
539 RCL 36
540 *
541 RCL 32
542 RCL 38
543 *
544 +
545 RCL 33
546 RCL 37
547 *
548 -

549	RCL	25
550	*	
551	-	
552	RCL	23
553	RCL	36
554	*	
555	RCL	28
556	RCL	37
557	*	
558	+	
559	RCL	27
560	RCL	38
561	*	
562	-	
563	RCL	30
564	*	
565	-	
566	RCL	19
567	*	
568	ST+	51
569	RCL	61
570	RCL	21
571	*	
572	ST+	51
573	RCL	72
574	RCL	26
575	*	
576	RCL	31
577	RCL	39
578	*	
579	RCL	34
580	RCL	36
581	*	
582	-	
583	ST0	75
584	RCL	28
585	*	
586	-	
587	RCL	31
588	RCL	38
589	*	
590	RCL	33
591	RCL	36
592	*	
593	-	
594	ST0	76
595	RCL	29
596	*	
597	+	
598	RCL	22
599	*	
600	ST -	51
601	RCL	73
602	RCL	26
603	*	
604	RCL	75
605	RCL	27
606	*	
607	-	
608	RCL	31
609	RCL	37
610	*	
611	RCL	32
612	RCL	36
613	*	
614	-	
615	ST0	77
616	RCL	29
617	*	
618	+	
619	RCL	23
620	*	
621	ST+	51
622	RCL	74
623	RCL	26
624	*	
625	RCL	76
626	RCL	27
627	*	
628	-	
629	RCL	77

630 RCL 28
631 *
632 +
633 RCL 24
634 *
635 ST- 51
636 LBL 52
637 RCL 78
638 RCL 27
639 +
640 STO 82
641 RCL 21
642 *
643 RCL 81
644 -
645 RCL 24
646 RCL 36
647 *
648 -
649 STO 65
650 RCL 60
651 +
652 RCL 15
653 *
654 CHS
655 STO 52
656 RCL 82
657 RCL 20
658 *
659 RCL 22
660 RCL 25
661 *
662 -
663 RCL 23
664 RCL 30
665 *
666 -
667 RCL 24
668 RCL 35
669 *
670 -
671 RCL 16
672 *
673 ST+ 52
674 RCL 78
675 RCL 21
676 +
677 RCL 25
678 *
679 RCL 20
680 RCL 26
681 *
682 -
683 RCL 69
684 -
685 RCL 17
686 *
687 ST+ 52
688 RCL 79
689 RCL 21
690 +
691 RCL 30
692 *
693 RCL 20
694 RCL 31
695 *
696 -
697 RCL 70
698 -
699 RCL 18
700 *
701 ST+ 52
702 RCL 80
703 RCL 21
704 +
705 RCL 35
706 *
707 RCL 20
708 RCL 36
709 *
710 -

711	RCL	71
712	-	
713	RCL	19
714	*	
715	ST+	52
716	RCL	60
717	RCL	21
718	*	
719	ST-	52
720	RCL	62
721	RCL	22
722	*	
723	ST+	52
724	RCL	63
725	RCL	23
726	*	
727	ST+	52
728	RCL	64
729	RCL	24
730	*	
731	ST+	52
732	RCL	61
733	ST-	52
734	LBL	53
735	RCL	82
736	RCL	21
737	+	
738	ST0	83
739	RCL	15
740	*	
741	RCL	65
742	+	
743	RCL	60
744	+	
745	RCL	16
746	RCL	20
747	*	
748	-	
749	RCL	17
750	RCL	25
751	*	
752	-	
753	RCL	18
754	RCL	30
755	*	
756	-	
757	RCL	19
758	RCL	35
759	*	
760	-	
761	ST0	53
762	LBL	54
763	RCL	83
764	RCL	15
765	+	
766	CHS	
767	ST0	54
768	LBL	50
769	SF	02
770	XR0M	01,06
771	CHS	
772	ST0	00
773	RCL	51
774	ST0	01
775	RCL	52
776	ST0	02
777	RCL	53
778	ST0	03
779	RCL	54
780	ST0	04
781	1	
782	ST0	05
783	LBL	06
784	RCL	59
785	ST0	06
786	ST0	22
787	2	
788	ST+	06
789	LBL	07
790	DSE	06
791	GT0	08

```
792 SF 00
793 SF 21
794 XROM 01,12
795 RTN
796 LBL 08
797 RCL 06
798 1
799 -
800 "a"
801 FIX 00
802 CF 29
803 ARCL X
804 >"="
805 FIX 04
806 SF 29
807 ARCL IND X
808 PROMPT
809 GTO 07
810 LBL 00
811 LBL 01
812 "N\1D2,3,4,5"
813 AVIEW
814 PSE
815 "XEQ MATH:MATRIX"
816 >" 1ST"
817 AVIEW
818 END
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

GAMMA and GAMMLN: Fast, Accurate Lanczos Approximation for Gamma Functions

Posted by [Steven Thomas Smith](#) on 1 Sept 2009, 12:49 p.m.

This code computes the Gamma and log-Gamma functions, $\Gamma(x)$ and $\log|\Gamma(x)|$, over all x with full or nearly full 10 digit accuracy. It uses [Lanczos's approximation](#), which is amazing: very high accuracy $\Gamma(x)$ with a couple of LN calls and a handful of arithmetic operations. The [Lanczos coefficients](#) for ($\gamma = 5$, $N = 6$) from [Numerical Recipes in C](#), 2d ed. are used. The Lanczos coefficients are stored 3 at a time in the synthetic MNO status registers, and simple flag setting is used for the fast two-pass loop. The excellent SANDMATH'-5 packages's compiled M-code is only about 40% faster. (This may suggest that SANDMATH might consider using a fast implementation of Lanczos's approximation, which would provide both compiled $\Gamma(x)$ and $\log|\Gamma(x)|$ within SANDMATH.) Lanczos effectively turns GAMMA and GAMMLN into fast system calls like LN and SIN.

Test case:

```
.5 XEQ GAMMA -> 1.772453851 == (-1/2)! == Sqrt(PI) [in 3.1 s on i41CX+'s
ridiculously slow "Default Speed"]
.5 XEQ SANDMATH-5 GAMMA -> 1.772453850 == Sqrt(PI) - 1e-9 [in 1.9 s]
```

The synthetic text below may be generated using the PPC ROM's "LB" load bytes function with these hex codes:

```
Line 57: FE 95 39 52 39 38 59 94 01 20 86 50 97 49 97
Line 58: F8 7F 91 23 17 39 57 20 00
Line 86: FE 02 40 14 09 82 40 01 98 65 05 32 03 30 01
Line 87: F8 7F 07 61 80 09 17 30 01
Line 93: F7 09 18 93 85 33 29 99
```

The explanation (see *Synthetic Programming Made Easy*, p. 40f.) for these byte codes are that the 6 Lanczos coefficients $c_1..c_6$ are stored in the HP41 as:

```
76.18009172947146 = +7.618009173 E1 = 07 61 80 09 17 30 01
-86.50532032941677 = -8.650532033 E1 = 98 65 05 32 03 30 01
24.01409824083091 = +2.401409824 E1 = 02 40 14 09 82 40 01
-1.231739572450155 = -1.231739572 E0 = 91 23 17 39 57 20 00
1.208650973866179e-3 = +1.208650974 E-3 = 01 20 86 50 97 49 97
-5.395239384953e-6 = -5.395239385 E-6 = 95 39 52 39 38 59 94
```

This code is generated by Antonio Lagana's i41CX+ iPhone emulator, with editing for the synthetic text lines. (Antonio is upgrading i41CX+ in v. 3.2 to do synthetic text viewing automatically.)

```
01 LBL "GAMMA"
02 CF 05
03 CF 07
04 X<0?
05 SF 07
06 ENTER
07 INT
08 2
09 /
10 FRC
```

```
11 X!=0?
12 CF 07
13 RDN
14 ENTER
15 FRC
16 CF 06
17 X=0?
18 SF 06
19 RDN
20 FC?C 06
21 GT0 01
22 E
23 -
24 FACT
25 RTN
26 LBL "GAMMLN"
27 SF 05
28 LBL 01
29 CLA
30 E
31 X<>Y
32 GT0 03
33 LBL 02
34 ENTER
35 ABS
36 LN
37 ST- N
38 RDN
39 E
40 +
41 LBL 03
42 X<Y?
43 GT0 02
44 ENTER
45 ENTER
46 4.5
47 +
48 ST- N
49 LN
50 RCL Y
51 .5
52 -
53 *
54 ST+ N
55 CLX
56 RCL N
57 "\95\39\52\39\38\59\94\01\20\86\50\97\49\97"
58 >"\91\23\17\39\57\20\00" ; hex, -5.395239385 E-6->0, +1.208650974 E-3->N, -1.231739572 E0-M
59 .
60 RCL Z
61 5
62 +
63 SF 06
64 LBL 04
65 X<> 0
66 RCL 0
67 /
68 +
69 E
70 ST- 0
71 X<> N
72 RCL 0
73 /
74 +
75 RCL N
76 ST- 0
77 X<> M
78 RCL 0
79 /
80 +
81 FC? 06
82 GT0 05
83 RCL 0
84 RCL M
85 -
86 "\95\39\52\39\38\59\94\01\20\86\50\97\49\97"
87 >"\07\61\80\09\17\30\01" ; hex, +2.401409824 E1->0, -8.650532033 E1->N, +7.618009173 E1->M
88 FS?C 06
89 GT0 04
90 LBL 05
91 LN1+X
```

```
92 +
93 "09\18\93\85\33\29\99" ; hex, LN(SQRT(2*PI)) = 9.189385332 E-1->M
94 RCL M
95 +
96 FS?C 05
97 RTN
98 E^X
99 FS?C 07
100 CHS
101 END
```

Edited: 25 Sept 2009, 12:00 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

Enhance your HP-97 -- Add a Clock and More

Posted by [Katie Wasserman](#) on 1 Sept 2009, 6:53 p.m.

The HP-97 and HP-97S are identical except for the addition of the I/O pod on the 97S. This project demonstrates how it is possible to achieve some of the functionality of the 97S (including the pod) on a plain HP-97. In particular, it demonstrates how input can be feed to a 97 in the same way that the 97S pod does, only more so....

Control Lines:

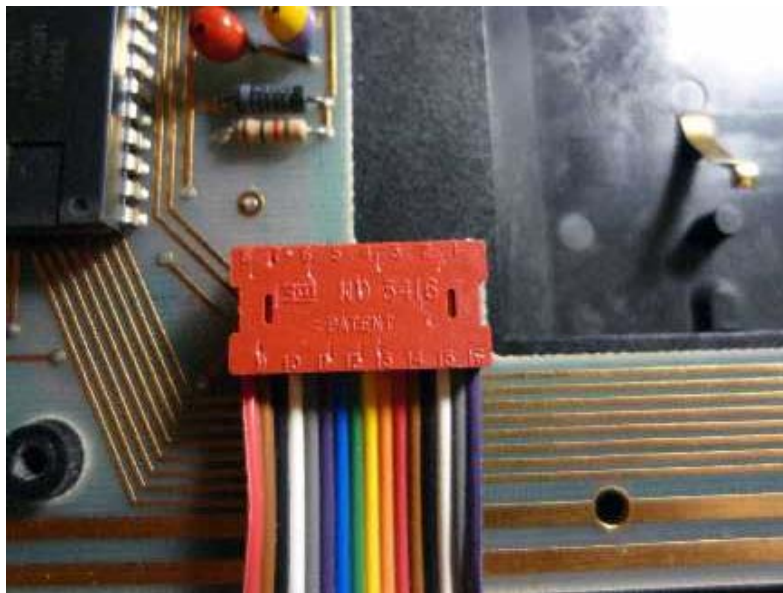
disclaimer: The control lines from an HP-97S to it's pod are only partially documented in the HP-97 service manual and the HP-97S Operation Guide. So, much of the following is from a bit of reverse engineering and therefore subject to inaccuracies.

On the HP-97S and newer HP-97 calculators all the control lines needed for the pod interface are conveniently brought to a 16 pin DIP socket location on the main logic board, but they exist on the earlier model HP-97 calculators as well, they are just not so conveniently accessed.

These are all the control lines along with their pin assignments on the DIP header:

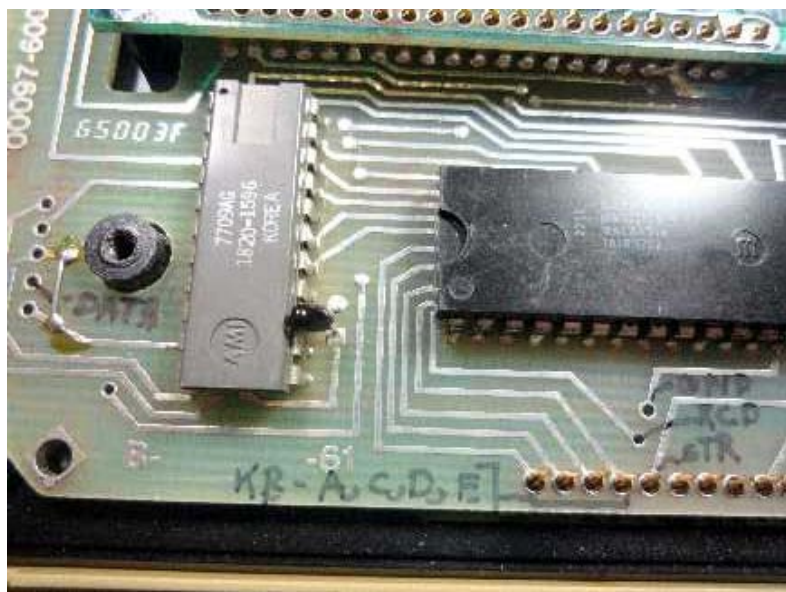
Control Line	Header Pin #
GND	1
n/c	2
phi -1	3
phi -2	4
RCD	5
KBA	6
KBC	7
KBD	8
+V	9
+V	10
STR	11
KBE	12
DATA	13
SYNC	14
IS/IA	15
GND	16

Here's a picture of the main board in the 97S (and newer 97). Note the ribbon cable that has replaced the standard cable that attaches to the pod and the closeup show the pin numbers on the IDC DIP header



Here's a picture of the main board in an older 97 and a closeup show the locations of the needed control lines for this project.





How the Pod Interface Works:

Inspection of the Keyboard PCA Schematic Diagram in the HP-97 service manual shows that a 1-of-14 counter/decoder chip (labeled as the cathode driver since it serves that purpose too) is used to scan the rows of the keyboard matrix. These 14 rows times the 4 columns (KBA, KBC, KBD, KBE) account for all 64 keys on the HP-97. The 14 outputs from the cathode driver chip are normally high and only go low when the row is being scanned. If a key on that row is pressed it will complete the circuit to one of the 4 column lines. So, pulling a column line low at the moment when the needed row is selected is equivalent to pressing the corresponding key. In this way, keyboard input can be simulated by an external circuit. The cathode driver chip is clocked by the rising edge of STR (strobe?) line and reset by the falling edge of RCD (reset cathode driver?) line. Knowing this it is straightforward to program a microcontroller chip to simulate key presses using these 6 control lines.

The 97S pod uses all of the lines coming out of the DIP header, why? The DATA, IS/IA (serial address), clock lines (phi-1 and phi-2) and SYNC lines are needed to capture the commands that the HP-97S is executing. Actually, the 97S only looks for 8 commands: Set Flag 0, Clear Flag 0 and the same for flags 1, 2 and 3. It uses these commands to set/clear 4 output lines on its 50-pin connector and to indicate that the calculator is ready for input from the user. This project makes no attempt to decode the address and data lines on the 97 and therefore has no flag output lines. However there is a simple way that the calculator can indicate that it is ready for input.

As it turns out, the DATA line has nearly continuous activity on it except when the number the calculator is displaying has a value of zero in the X register and is awaiting keyboard input. In this state, there are no signals on the DATA line (but there is a lot of noise for some reason -- a Schmitt trigger input can be used to overcome this problem). Thus a program running on the 97 can

simply clear the X register and execute a R/S command to indicate to an external microcontroller-based interface that it's ready for input. The external processor can then key in the numeric input followed by a R/S command to resume the program.

The Project:

This project makes use of all of the above findings to enhance the basic HP97. It's composed of a simple microcontroller, a Parallax SX28 with just 2K of ROM and 136 bytes of RAM, along with a DS1302 real time clock chip that is used for timekeeping and non-volatile memory. The SX not only simulates key presses on the 97 it also reads the keystrokes that the user presses on the 97. This is how the user tells the external microcontroller what functions to preform (e.g., set the time on the clock or generate a random number). Taken together -- 97 keyboard used to control the SX, SX used to control the 97 -- some useful enhancements can be implemented.

Since all the keys on the 97 do something, there needs to be a way to talk to the SX without the 97 doing anything, at least anything critical. There are several commands that don't make much sense in non-programming mode, for example $x<0?$ and DEL, these are used with the additional precaution of a double press of the 'f' key to prefix them.

Here's the complete list of commands to talk to the SX microcontroller that are implemented in this project:

f,f,7 -- start the continuously updating real time clock display (show: HH.MM SS)

f,f,1 -- stop the clock display

f,f,4 -- start the by request (CLX, R/S) updating real time clock entry
(entry will be: HH.MMSS, R/S)

f,f,+ -- set the clock. The display is cleared and the user needs to enter HH.MMSS
(24-hour format -- enter leading zeros)

f,f,- -- set the clock to 12 hour display format

f,f,x -- set the clock to 24 hour display format

f,f,8 -- set the number of seconds between display updates when the
continuous clock is running. The display is cleared and the user needs to
enter the number of seconds SS (enter a leading zero if needed).

f,f,RND -- generate a random number from 0-1 in the X register.

f,f,DSP -- show the scanning col.row of each subsequent keystroke.
This is useful for making further enhancements and saves time rather
than trying to read this from the service manual.

f,f,5 -- macro record. This will capture the next sequence of
keystrokes (up to 26) and save them in non-volatile memory. To stop
recording turn off the calculator. When the calculator is switched
back on the keystrokes will be replayed. This is useful for storing
often used constants in particular registers, setting default decimal
places, etc.

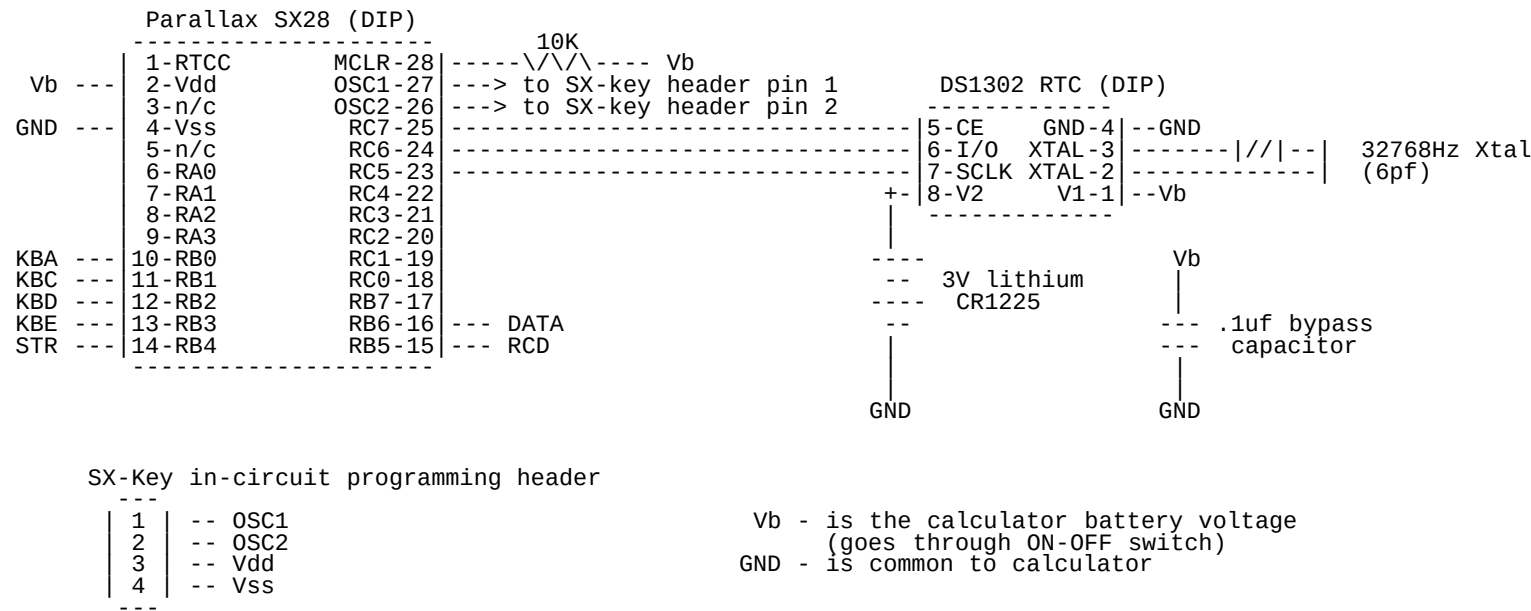
note: Execution of each command occurs when the key pressed is released. When the clock is displayed and changing rapidly it is necessary to press and hold 'f' and command keys for 1 second or so to overcome any interference from the microcontroller trying to update the display.

The electronics of this project are simple, just the SX28, DS1302, a 32768Hz clock crystal and a small lithium battery for the DS1302. All the work is done in the firmware. The SX28 microcontroller [source for this project](#) is written in a BASIC-like language called SX/B that Parallax makes freely available. It's an extremely simple, compiled language with some built-in speciality commands that came from Parallax's Basic Stamp. The compiler is so simple that only a single arithmetic operation can be done in an assignment statement. If you look through the code you'll see all sorts of unreasonable things that are there just to fit this amount of stuff into 2K of ROM and to deal with the compiler's limitations. I'm certainly no expert SX/B programmer so I'm sure there's room for improvement. It's not a pretty language nor a pretty program but hopefully the code is readable enough.

I choose the SX chip and SX/B just because I had them around and had a bit of experience using them. A better choice might have

been to use a microcontroller with a built-in RTC, more memory, etc.

The schematic is simple enough for character drawing:



Here's a picture of the prototype and how it fits inside the 97 case. There's a perfect space for it just above battery compartment and it mounts directly onto the exposed ends of the screws holding the security tab in place.



I've made a little video demonstrating the features. You'll see that this is a 97S but it had no pod so it's the same as a regular 97 only with the convenience of a 16 pin DIP header to access the needed control lines -- the missing pod was the impetus for this project.

[update: May 10, 2010] Etienne Victoria was successful in building this project and installing it in an HP-97 with the old style main circuit board. [Pictures are here.](#)

[YouTube Video Demo](#)

Much more can be done this, particularly if a different microcontroller with more memory is used. It would be easy to add a serial connection between a PC and the SX and use the PC to download programs into the 97. Digital and analog input from sensors can be feed via the SX to the 97. Many more commands can be added to the 97 using the 'f-f' prefix. The keystroke macro facility can be easily expanded to do all sorts of stuff, etc..

Edited: 11 Oct 2014, 2:02 p.m.

[Edit Message](#)

[Delete Message](#)

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]
[Previous | Next]

HP41 Prime Factorization Revisited

Posted by [Ángel Martín](#) on 27 July 2009, 3:58 a.m.

As part of the SandMath Module improvements I decided to enhance the Prime factorization program(s).

I came up with two versions, the first one only uses ALPHA to show the results - similar to previous versions of the program. Using the MCODE function PRIME? speeds up the execution substantially, not that it matters a lot (specially when used an emulated 41 version) but anyway.

But the second (and more interesting) one stores the prime factors (and their exponents) into a Matrix, so that the original number can be rebuilt. Such matrix is dynamically redimensioned as new prime factors are obtained, which adds a little extra execution time - but it's worth it.

With just a few more program steps it also calculates the Euler function of the given number, taking advantage of having all the prime factors conveniently stored in the matrix. Easy when things are properly structured, it'd seem.

Here they are - Version #2 made it into the SandMath module, but you can choose the better one for you :-)

1 LBL "PRMF"	1 LBL "PRMF"	1 LBL "EULER"
2 CF 00	2 "PRMF"	2 XROM "PRMF"
3 INT	3 1,002	3 0
4 ABS	4 MATDIM	4 MSIJ
5 CLA	5 CLX	5 X<>Y
6 AIP	6 MSIJ A	6 LBL 07
7 "@="	7 RDN	7 MRC+
8 PRIME?	8 CF 00	8 1/X
9 SF 00	9 INT	9 CHS
10 AIP first pf	10 ABS	10 INCX
11 X=1?	11 PRIME?	11 *
12 GT0 01	12 SF 00	12 FC? 09
13 FS?C 00 we're done	13 MSR+	13 GT0 07
14 GT0 01	14 X=1?	14 END
15 STO 01 save grouping	15 GT0 01	
16 ST/ L	16 FS?C 00	
17 LASTX reduced number	17 GT0 01	
18 LBL 05	18 STO 01	
19 E	19 ST/ L	
20 STO 00 reset counter	20 LASTX	
21 RDN	21 LBL 05	
22 LBL 00	22 E	
23 RCL 01 previous pf	23 STO 00	
24 X<>Y	24 RDN	
25 PRIME?	25 LBL 00	
26 SF 00	26 RCL 01	
27 X#Y? different PF?	27 X<>Y	
28 GT0 02 YES	28 PRIME?	
29 ISG 00 increase count	29 SF 00	
30 NOP SAME pf	30 X#Y?	
31 FS?C 00 was it prime?	31 GT0 02	
32 GT0 03 skip if Prime	32 ISG 00	
33 ST/ L	33 NOP	
34 LASTX reduced number	34 FS?C 00	
35 GT0 00	35 GT0 03	
36 LBL 03 Prime found	36 ST/ L	
37 RCL 00	37 LASTX	

38 X=1?
39 GTO 01
40 "@^"
41 AIP
42 GTO 01
43 LBL 02
44 STO 01
45 RCL 00
46 X=1?
47 GTO 03
48 "@^"
49 AIP
50 LBL 03
51 RDN
52 "@*"
53 AIP
54 FS?C 00
55 GTO 01
56 ST/ L
57 LASTX
58 GTO 05
59 LBL 01
60 AVIEW
61 END

NEW pf

38 GTO 00
39 LBL 03
40 RCL 00
41 MSR+
42 GTO 01
43 LBL 02
44 STO 01
45 RCL 00
46 MSR+
47 RDN
48 ST/ L
49 LASTX
50 DIM?
51 INCX
52 MATDIM
53 X<> Z
54 MSR+
55 FS?C 00
56 GTO 01
57 X<>Y
58 GTO 05
59 LBL 01
60 E
61 FC? 10
62 MSR+
63 STO 00
64 MSIJ
65 CLA
66 LBL 06
67 MRR+
68 AIP
69 MRR+
70 X=1?
71 GTO 04
72 "@^"
73 AIP
74 LBL 04
75 Y^X
76 ST* 00
77 FC? 10
78 "@*"
79 FC? 10
80 GTO 06
81 RCL 00
82 PROMPT
83 END

all done
initiate R00
to rebuild the number
and display factors

Edited: 12 Aug 2009, 1:50 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

Mental juggling: Calculating the day of week

Posted by [Thomas Klemm](#) on 7 July 2009, 1:36 p.m.

My father has a quirk: When he meets someone new, he asks after the date of birth and then after a few seconds of reflection, he says the corresponding day of the week.
I do not know exactly how he makes, but he knows many of the weekday data from memory. This is used to calculate the difference to this particular day.
Though there are algorithms to do that with a computer or a programmable calculator I was trying to find a way to do it in my mind. However mental arithmetic is not one of my skills.

Let me explain it first with an example: 22.06.1955

Day22->1

Month6->4

Year55->5

10->3->Wednesday

Day

This one is easy: it's calculating the remainder after division by 7.

Example: $22 = 3 * 7 + 1$

Actually you don't have to divide by 7. You just need to know the closest multiple of 7 which is smaller or equal and then calculate the difference:

$22 - 21 = 1$

Month

Here you don't have to calculate anything but keep a table in your mind:

Month	1	2	3	4	5	6	7	8	9	10	11	12
Magic number	1	4	3	6	1	4	6	2	5	0	3	5

As I can't remember such a table very well I rearranged it a little:

Month	Magic number											
1	2	3	1	4	3							
4	5	6	6	1	4							
7	8	9	6	2	5							
10	11	12	0	3	5							

For Januray and March the mapping is identity, for October it is 0 which means it can be ignored. Then you can see that the 2nd and 3rd line both start with 6. In the 2nd line 1 4 from the first line is repeated but shifted by 1 whereas in

the 3rd line 1 is added giving 2 5. Well, the last line: 10 is 0 and 1 added to 2 gives 3 and 5 is kept. But I'm sure you'll find a different way to memorize this table.

How did I calculate this mapping?

Starting with March which is set to 3 either 31 or 30 is added depending on how many days the previous month has:

month	#	days	sum	% 7
3		31	3	3
4		30	34	6
5		31	64	1
6		30	95	4
7		31	125	6
8		31	156	2
9		30	187	5
10		31	217	0
11		30	248	3
12		31	278	5
1		31	309	1
2		*	340	4

(*) Since February is the last month in the list we don't have to take leap years into account here. This is the reason why we started with March.

Year

Leap years make things complicated. Here again we have a mapping table:

0	:	00	06		17	23	28	34		45	51	56	62		73	79	84	90		101	107
1	:	01	07	12	18		29	35	40	46		57	63	68	74		85	91	96	102	
2	:	02		13	19	24	30		41	47	52	58		69	75	80	86		97	103	108
3	:	03	08	14		25	31	36	42		53	59	64	70		81	87	92	98		109
4	:		09	15	20	26		37	43	48	54		65	71	76	82		93	99	104	110
5	:	04	10		21	27	32	38		49	55	60	66		77	83	88	94		105	111
6	:	05	11	16	22		33	39	44	50		61	67	72	78		89	95	100		106

Search the year (- 1900) in the table and find the corresponding number in the leftmost column.

Now this is something I don't want to learn by heart. However you may notice that the pattern repeats after 28 years so we just need to remember the first 28 years. Unfortunately calculating modulo 28 isn't something I can do easily in my mind.

This is the easiest way I came up to calculate the number:

1. Find the closest multiple of 4 smaller or equal to the year. Keep the difference.
2. Divide that multiple of 4 by 2.
3. Find the difference to the closest multiple of 7 bigger or equal to that number.
4. Add this to the difference from step 2.

Example: $55 = 52 + 3$
 $52 : 2 = 26$
 $26 + 2 = 28$
 $3 + 2 = 5$

As the calculations are carried out modulo 7 we can also subtract the difference to the closest multiple of 7 smaller than that number in steps 3 and 4.

Example: $46 = 44 + 2$
 $44 : 2 = 22$
 $22 - 1 = 21$
 $2 - 1 = 1$

Shortcut for years that are a multiple of 7:

1. Find the remainder after dividing by 4
2. Multiply that number by 5

$$\text{Example: } 49 = 48 + \frac{1}{1} * 5 = 5$$

$$91 = 88 + \frac{3}{3} * 5 = 15 = 1 \pmod{7}$$

Important note: You may have noticed that the mapping table for the months starts with March. Thus January and February belong to the year before the actual year. Therefore for these two months the year has to be reduced by 1 before any calculations.

I know there are different ways to do that calculation. But it turned out that I can calculate n modulo 4 quite fast while I have difficulties to divide a number by 4 if it's bigger than say 50. On the other I find no trouble to divide a number by 2. Probably it's all a question of training. However it seems I just don't do enough divisions by 12 to use [Lewis Carroll's method](#):

Quote:

Add together the number of dozens, the overplus, and the number of 4s in the overplus.

Day of the week

The following table is used to map the resulting number to the day of the week:

0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

Explanation

What we actually do is calculating the difference of the date to 28.02.1900 modulo 7. This interval is split into three parts from which modulo 7 can be found easier:

$$\begin{array}{rclcl}
 22.06.1955 & & & & \\
 0.06.1955 & 22 & = & 1 \pmod{7} & \\
 0.03.1955 & 31 + 30 + 31 = 92 & = & 1 \pmod{7} & \\
 0.03.1900 & 55 + 13 = 68 & = & 5 \pmod{7} & // 55 : 4 = 13 \text{ leap years} \\
 28.02.1900 & & = & 3 & \\
 & & & \underline{\quad} & \\
 & & & 10 & = 3 \pmod{7}
 \end{array}$$

Since 28.02.1900 was a Wednesday the mapping table of the months starts with 3.

What remains unexplained are the two formulas to calculate the mapping of the year:

$$\begin{aligned}
 \text{Let's assume } y &= 4n + r \quad ; \quad 0 \leq r < 4 \\
 \text{Then } d &= 365y + n = y + n \pmod{7} \quad // \text{ since } 365 = 1 \pmod{7} \\
 &= 4n + r + n = r + 5n = r - 2n \pmod{7} \\
 2n &= 4n / 2
 \end{aligned}$$

$$d = r - 4n / 2 \pmod{7} \quad // \text{ here we use that } 2 \mid 4n$$

Last not least the case where the year is a multiple of 7:

$$\begin{aligned}
 y &= 4n + r = 7k = 0 \pmod{7} \\
 r &= -4n = 3n \pmod{7} \quad // \text{ multiply both sides by 5}
 \end{aligned}$$

$$\begin{aligned}5r &= 15n = n \ (7) \\d &= y + n = 0 + 5r \ (7)\end{aligned}$$

Restriction

The method described here works only for dates from 01.03.1900 to 28.02.2100. However it could easily be adjusted.

Examples

6 July 1979

$$\begin{aligned}79 &- 76 = 3 \\76 &/ 2 = 38 \\38 &- 3 = 35 \\3 &- 3 = 0\end{aligned}$$

July -> 6

$$0 + 6 + 6 = 12 = 5 \ (7)$$

or shortened

$$\begin{array}{r}79 \ 76 \ 3 \\38 \ 35 \ -3 \\0 \ 6 \ -1 \ 5 \ \rightarrow \text{Friday}\end{array}$$

December 7th, 1941

$$\begin{aligned}41 &- 40 = 1 \\40 &/ 2 = 20 \\20 &+ 1 = 21 \\1 &+ 1 = 2\end{aligned}$$

Dec -> 5

$$2 + 5 + 7 = 14 = 0 \ (7)$$

or shortened

$$\begin{array}{r}41 \ 40 \ 1 \\20 \ 21 \ 1 \\2 \ 5 \ 0 \ \rightarrow \text{Sunday}\end{array}$$

20.02.1964

$$\begin{aligned}63 &- 60 = 3 \\3 * 5 &= 15 = 1 \ (7)\end{aligned}$$

Feb -> 4

$$1 + 4 + 20 = 25 = 4 \ (7)$$

or shortened

$$\begin{array}{r}63 \ 60 \ 3 \\15 \\1 \ 4 \ -1 \ 4 \ \rightarrow \text{Thursday}\end{array}$$

Programming

How about a training program for your favorite HP calculator that displays a random date and asks you to enter the appropriate day of the week?

Edited: 24 Sept 2010, 3:21 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

A non destructive (easy) method for opening a spice calculator.

Posted by [Geoff Quickfall](#) on 27 May 2009, 1:47 p.m.

Hello again,

To start, I hesitated to repair and restore my spice units until I read or developed a technique to separate the two calculator shells. I have seen described a pull apart method with a lot of force used and deformation to the polycarbonate case. Hence the following posting.

I have used this method succesfully on 5 out of 5 spices. It did take one almost destroyed unit to figure this out. Actually the unit came in a total mess and the only salvageable stuff was the LED panel (solderless), battery hatch and battery hatch ribbon.

Obviously HP had an easy way to get into a spice, apart from the pry and pull and push method that makes one very reticent to work on one of these.

After careful examination of the case I have created the following steps to get in.

Of course, the obvious, looking at the back of the calculator or bottom shell we see the battery hatch. Unlike other HP's there are no hidden screws (figure 1).

FIGURE 1



Remove the hatch and you will see two screws (figure 2), remove these.

FIGURE 2



Once the screws are removed, grasp the calculator and lift apart the two halves at the battery end (figure 3). The calculator will pivot around the foot of the two shells. Only pull the calculator apart as far as the figure indicates.

FIGURE 3



Now that the two parts are separated at the top of the calculator, insert a long and extremely thin knife blade into the seam

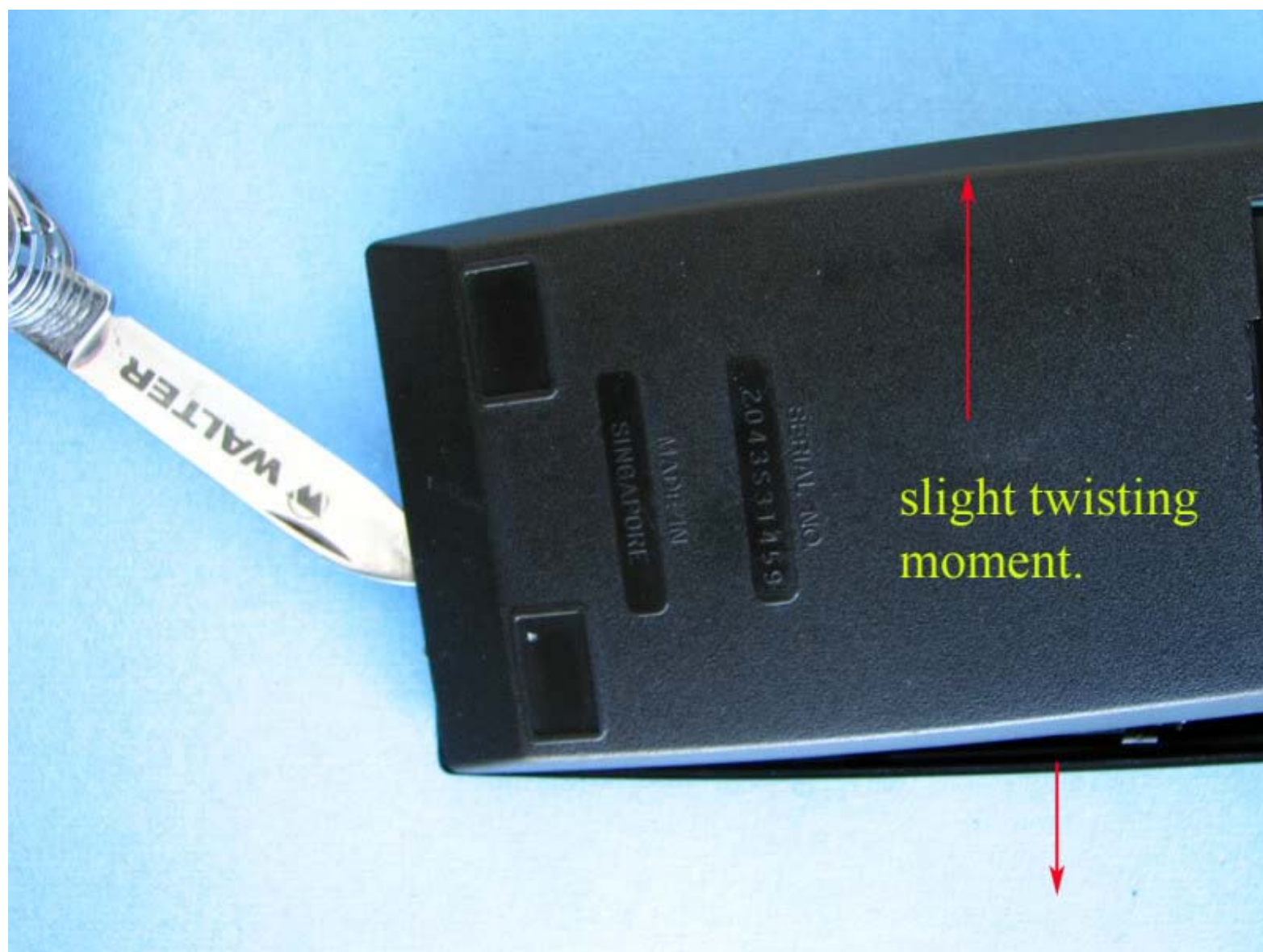
which separates the shells at the foot of the calculator (figure 4). The seam is vertical to the longitudinal plane of the calculator. That is, the bottom shell fits inside a lip of the top shell. The knife must be inserted vertically for a very shallow depth (1mm) and then turned horizontally. This has the affect of lifting the bottom shell out of the upper shell lip at the foot of the calculator. You will see why this is important soon.

FIGURE 4



Now that the bottom shell is above the lip of the top shell and while keeping the knife in place to keep the bottom shell out of the top shell recess, give the calculator shells a twist (figure 5). This is a horizontal twist; each shell in the opposite direction of the other. Be gentle as this method does not require force.

FIGURE 5



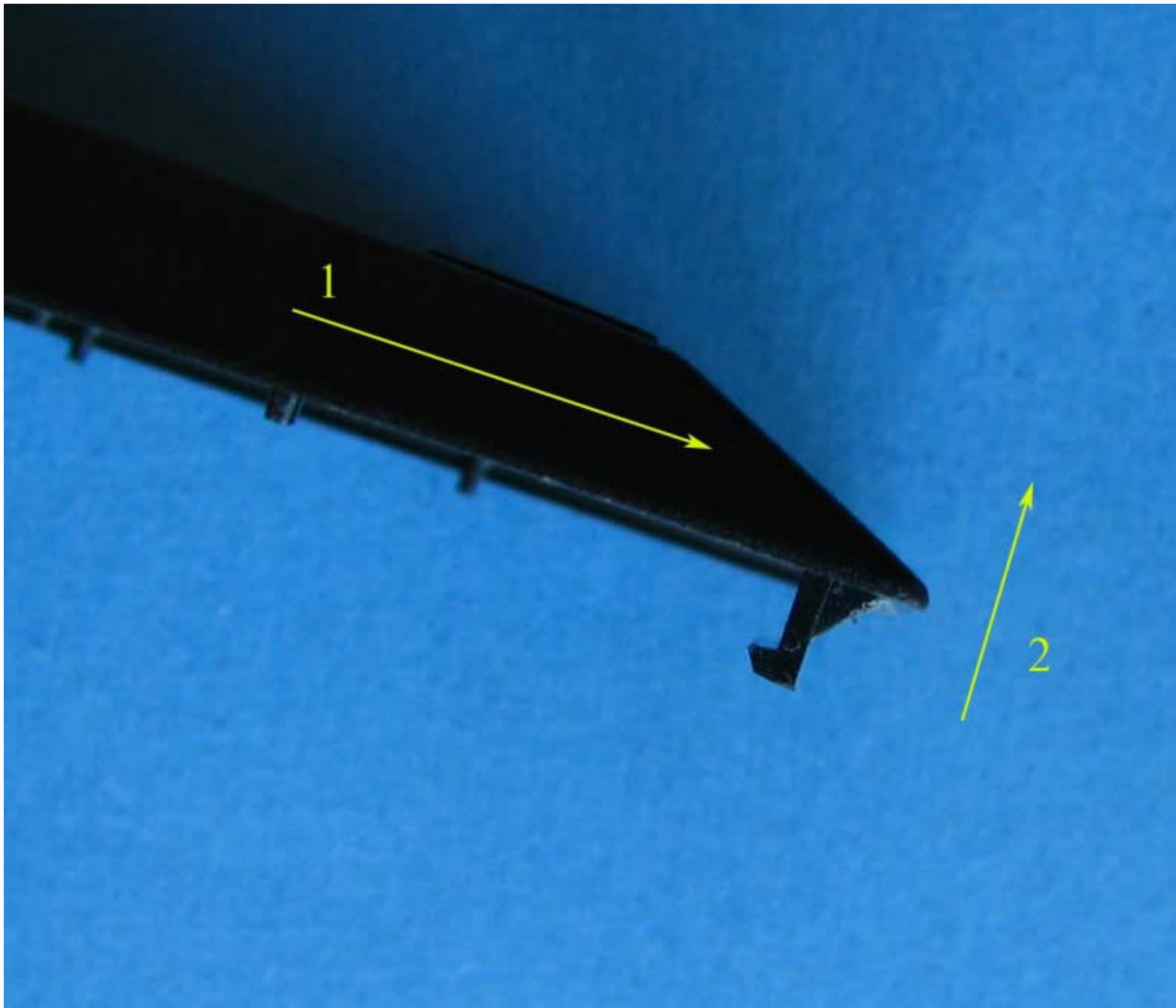
In figure 6 we can see that the bottom shell now protrudes past the top shell. Referring to figure 5 again, and while still applying a twisting moment as described, give the knife a gentle twist. This will pry apart the bottom shell and top shell. This combined with the twisting moment you are applying to the case will release the internal latch mechanism. The mechanism will release with a quiet snap.

FIGURE 6



In figure 7 you can see the L shaped latch which causes the headache, well, maybe not any more! Arrow 1 shows the direction of the twisting moment applies on the latch with the knife in place and the bottom shell raised out of the lip. Arrow two shows the final twisting moment of the kife releasing the two halves.

FIGURE 7



Of course it is difficult to describe but in summary:

1. remove screws
2. insert flat blade at foot of calculator to raise the bottom shell out of it's recess (lip) in the top shell
3. while the blade is in place and the bottom shell is raised; give the two shell a rotational twist which will raise the rest of the bottom shell out of the recess
4. give the knife a rotational twist while still applying the rotational twist to the two shells
5. you should hear a gentle satisfying snap as the L shaped latch separates from its anchor point.

It's that easy!!

Cheers, Geoff May 2009

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 41 Battery posts, another repair

Posted by [Geoff Quickfall](#) on 20 Apr 2009, 2:58 p.m.

I saw Nigel Bamber's posting earlier and coincidentally was in the process of repairing a CV myself.

Mine came with the x functions, x memory and a time chip with a really worn keyboard. I mean REALLY worn. Well thanks to Wlodek in England I secured a new looking keyboard with two cracked posts.

Heres how I went about the repair:

Firstly, I preassembled the new keyboard to the rest of the CV and had to use some finger pressure to ensure contact as two of the posts were cracked. I determined the 'on' rocker key was not working as well as the E+ and 1/X keys.

I removed and soaked the keyboard in a solution of liquid detergent and warm water. The keyboard was placed vertically with the LCD above the level of the water. I rinsed the keyboard with water, dilute vinegar, water and then 99% alcohol. The keyboard was placed in a drying box with the dryer set for ambient air temperature.

The calculator was again assembled and the keyboard tested perfectly for all key punches and rocker arm switch function.

Now to the posts:

Tools required:

1. liquid plastic cement and brush.
2. old piece of soft plastic (ex. memory module housing).
3. exacto knife.
4. silicon tubing or small o-rings (inner diameter smaller then outer diameter of post).

silicon tubing:	ID	2.55 mm
	OD	5.60 mm
		Fuel line hose for model aircraft. Specify
		military specs as the hole is dead centre and
		therefore usable for pinch roller
o-rings	ID	2.55 mm
	OD	6.50 mm
		Hardware store.

In figure 1, you can see the reverse side of the keyboard. The two posts on the right are cracked in two places, almost opposite of each other.

FIGURE 1

CRACKED POSTS ON A 41CV

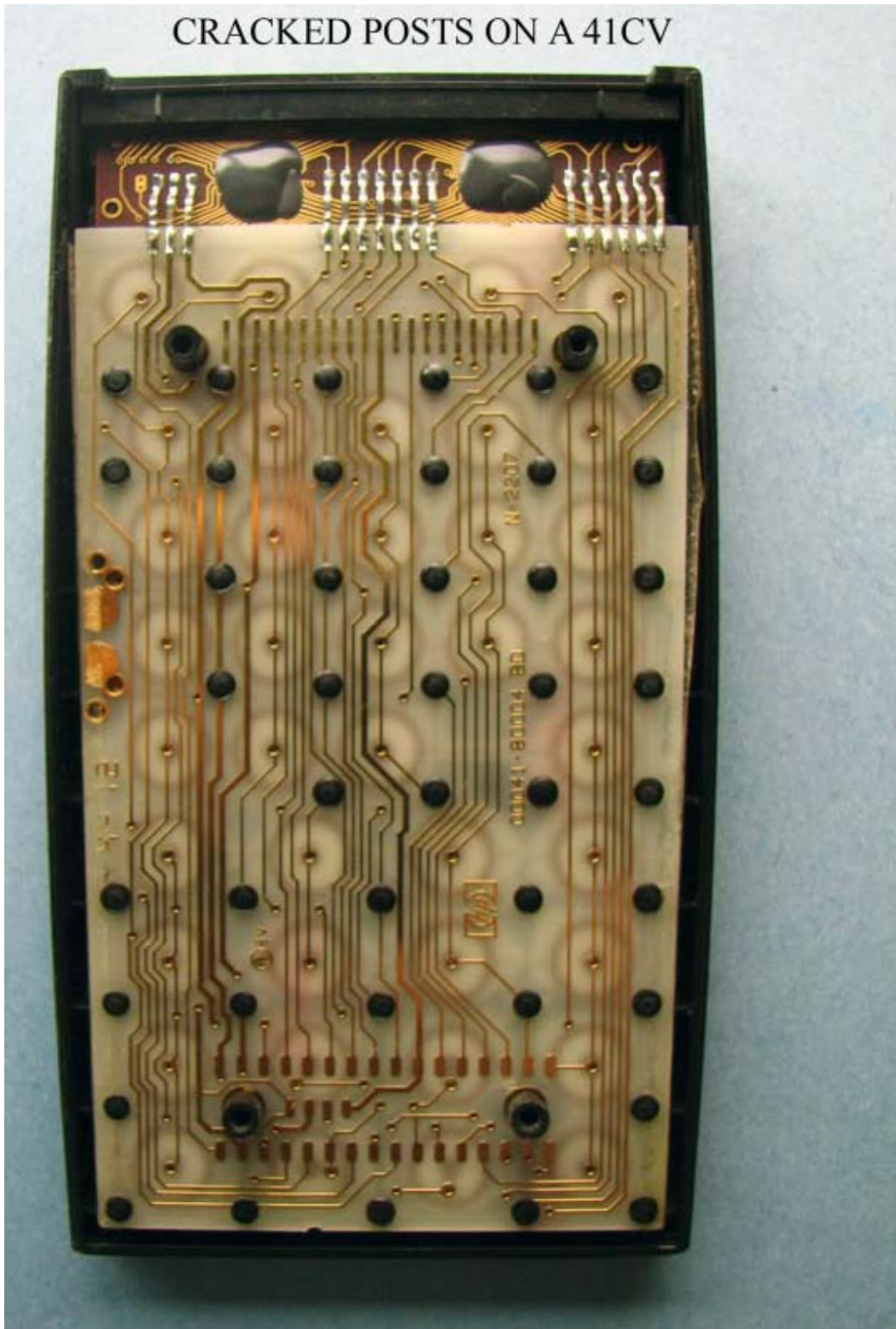
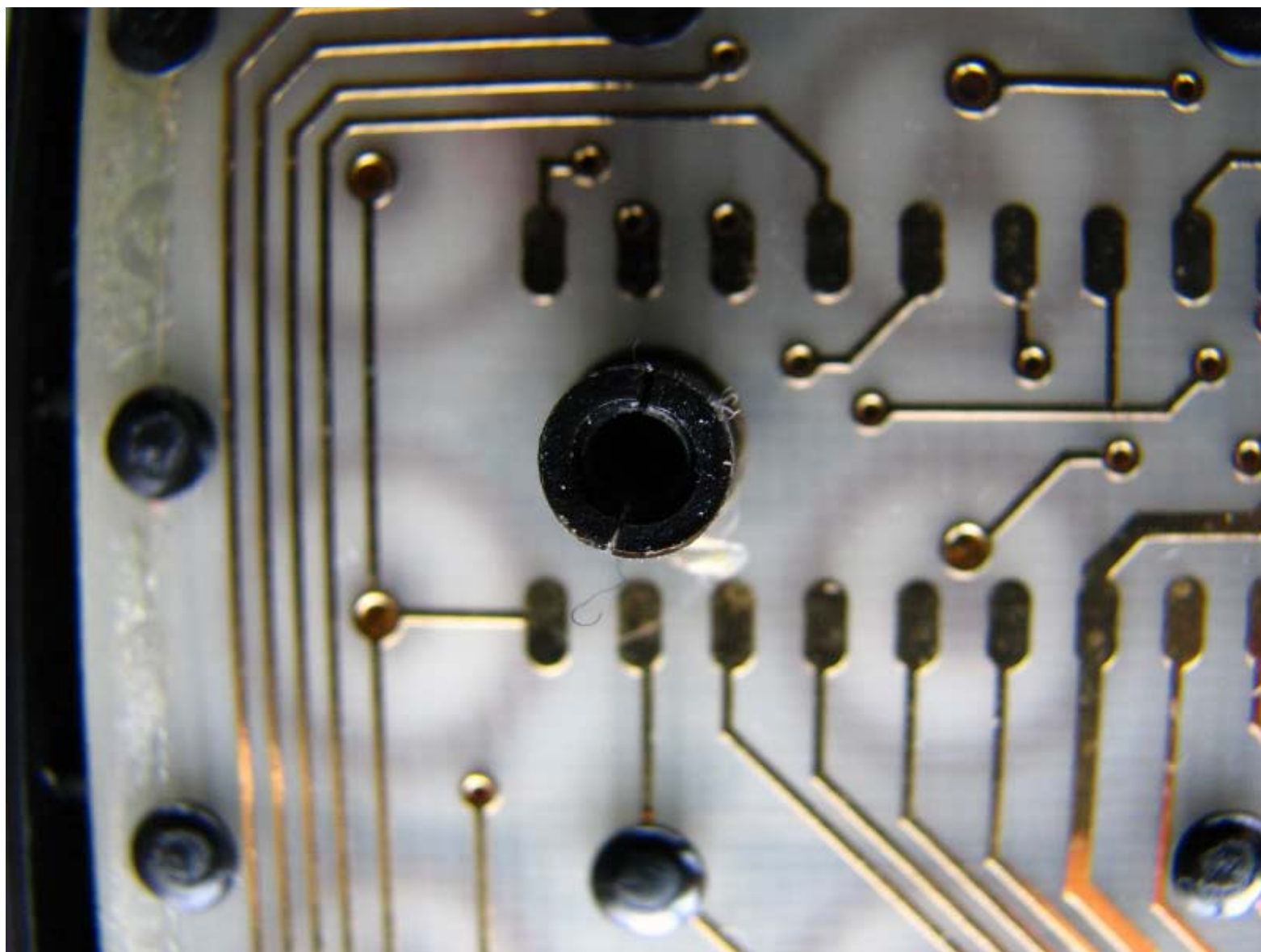


Figure 2 is a close-up of a typical cracked post, usually in two places nearly opposite to each other. The correct procedure for screwing an HP screw back into it's socket is to place the screw in the hole. Then reverse the screw motion, that is unscrew the loose screw until you feel it click. Then screw gently in the correct direction for tightening. By doing this you set the threads of the screw in synch with the threads of the post thereby not cross threading.

FIGURE 2



Once the cracked posts have been identified (or stripped posts as this routine works for these) a clamping system must be formulated. I used silicon tubing, the spare tube I have for fixing pinch rollers in HP card readers. Conversely the o-rings for the same purpose, suffice to say that the tube must have an inner diameter smaller than the outer diameter of the post (Figure 3).

FIGURE 3

SILICON TUBING; THE SAME USED IN REPAIRING
PINCH ROLLERS ON THE HP 41, 65, 67 & 97,



OR THE O-RINGS ALSO USED FOR PINCH ROLLER
REPAIRS.



Since the cracks are to be repaired we can assume that the internal threads of the posts are either stripped or damaged. Every time the screw enters and is removed it grates away at the internal threads of the post removing material and leading to a stripped post. I remedied this by securing a soft plastic material in the form of memory module housing. Shave off some of the plastic using an exacto knife. When the shavings flake off some of them form a perfect circle. I inserted these into the posts along with some straight flakes. This is done before the glue is applied. (figure 4)

FIGURE 4



Here you can see a series of photos where the cracks have been clamped together by the silicon tubing or the o-rings. Once these posts are clamped, add the shavings into the post.

FIGURE 5

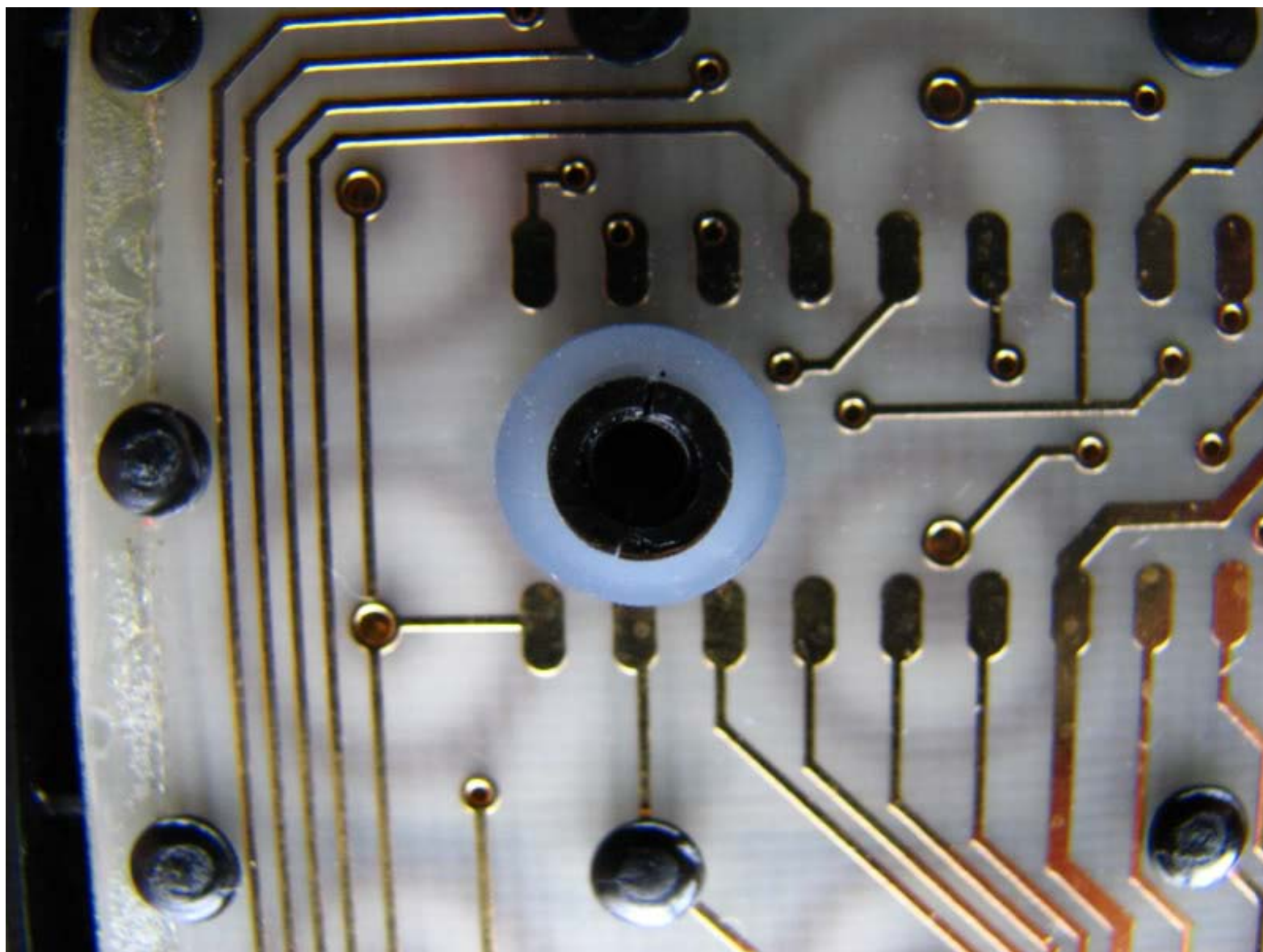


FIGURE 6

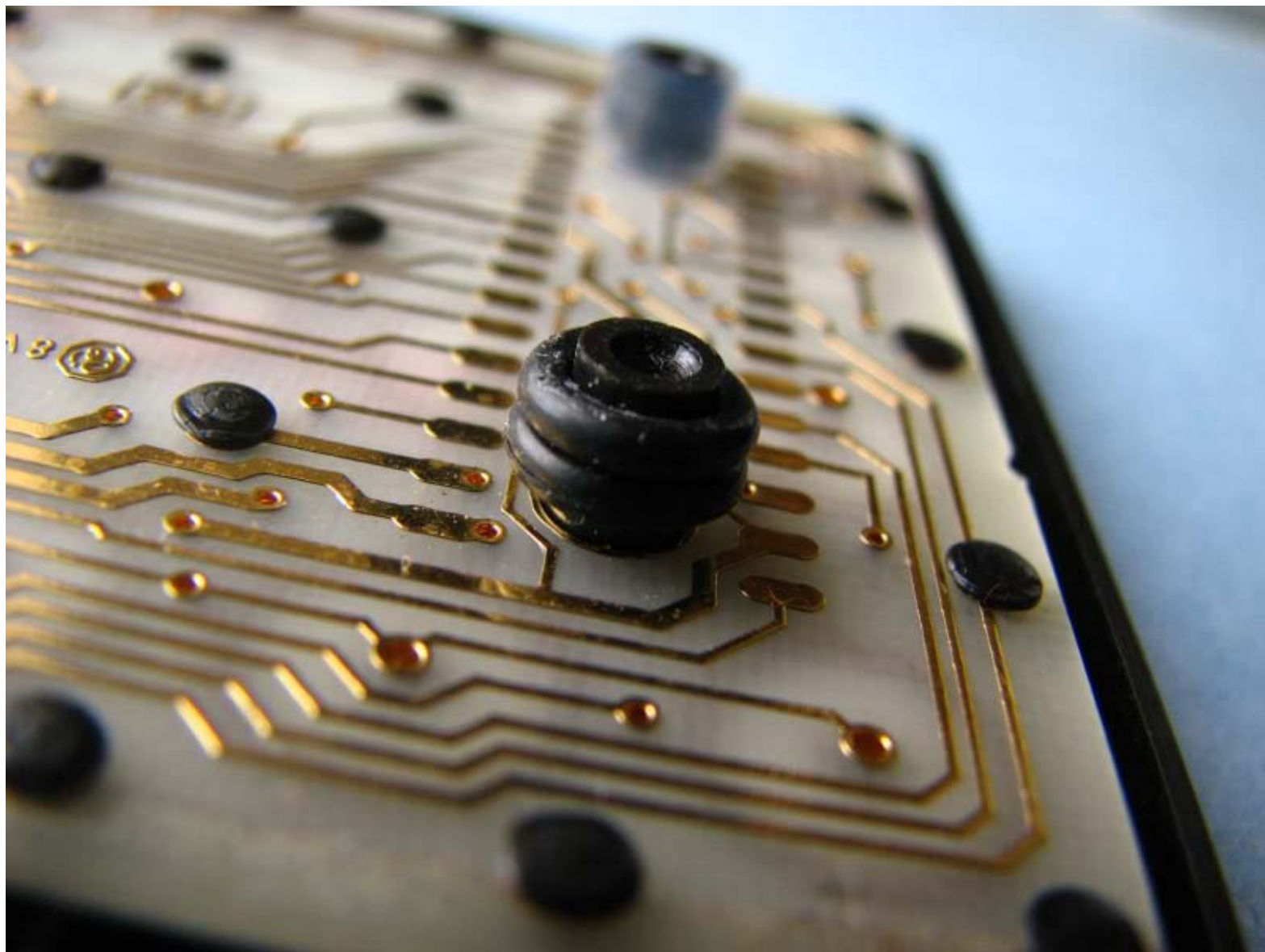
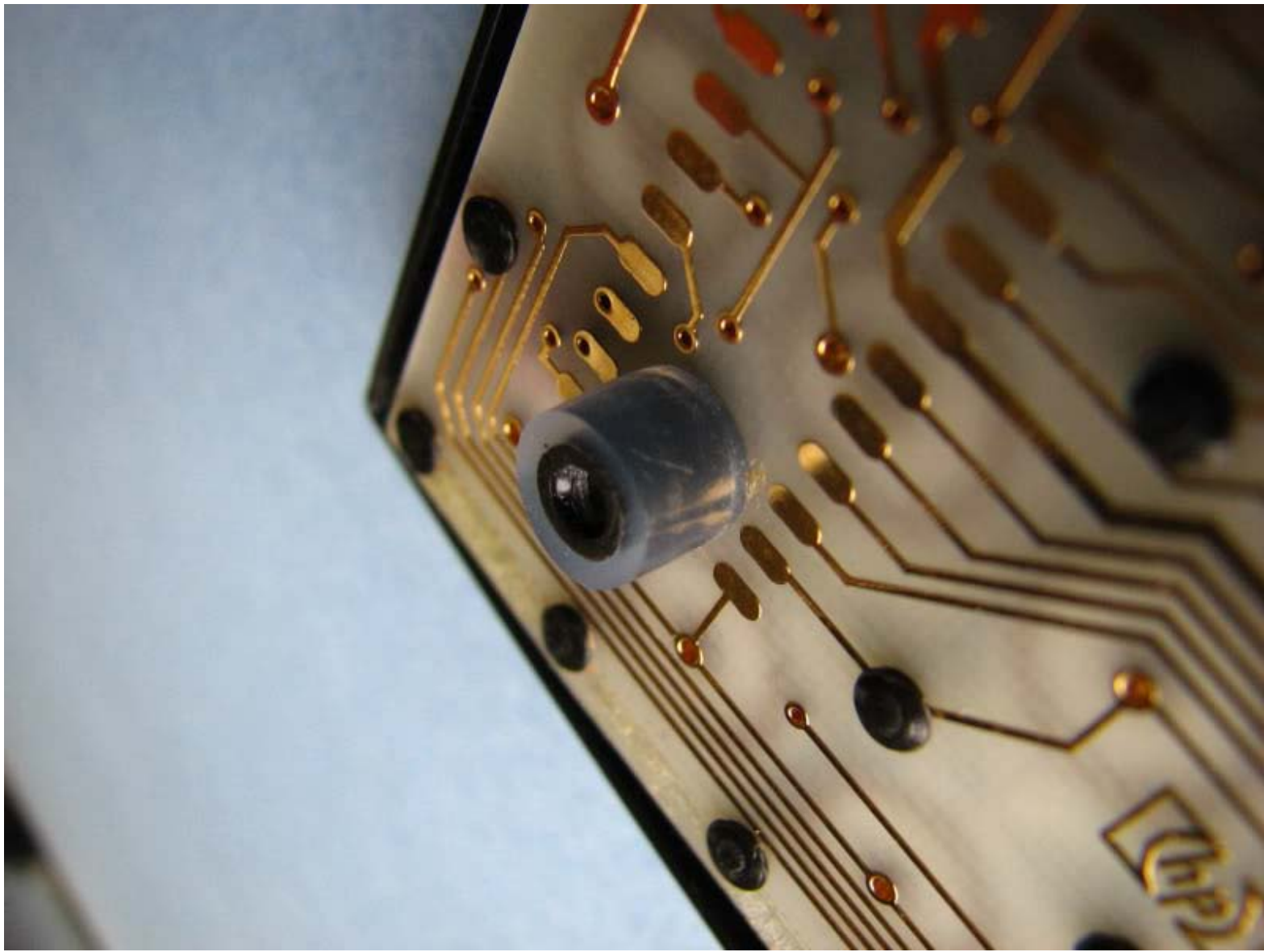


FIGURE 7



Using the liquid plastic cement, add a good dollop (a drop) to the inside of the clamped post with shavings added. The glue will seep into the cracks and soften the shavings. It will also bond the shavings to the posts, especially the cylindrical shavings added before hand. The clamps will insure the posts cracks have the requisite pressure exerted for glueing purposes.

FIGURE 8



Now let the post dry. I left it for a day; you certainly don't want to rush things.

As a last step I screwed the screws into the posts gently starting them as I outlined earlier. I also did this with the silicon clamps in place and the back of the case not attached. Do not screw all the way down but just start them until they bite. Since the back of the case is not in position the screws will be too long for the posts and you can damage the keyboard!

Now remove the clamps (silicon or o-rings) and replace the back. Using the screw technique mentioned above, tighten up the case.

It's that easy ;-)

Geoff

Edited: 21 Apr 2009, 1:51 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Power for HP82162A Printer and HP9114 Disc Drive

Posted by [Philip Isaac \(Canada\)](#) on 3 Mar 2009, 12:02 p.m.

When I found I could no longer get battery packs for the thermal printer and disc drive I have in the IL loop with my HP41CX I wired them to run off a 6 volt tractor battery (from my 1957 JD420 crawler).The system works well. The HP7470A Plotter, also in the loop, uses 110AC so doesn't need the battery connection.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

INIT-Module for ERAMCO RSU-2 Rambox

Posted by [PeterP](#) on 18 Feb 2009, 6:37 p.m.

Full Manual and Files

<http://www.hpmuseum.org/guest/peterp/initrsu.pdf>

<http://www.hpmuseum.org/guest/peterp/initrsu.zip>

Summary

Based on Diego Diaz's great article about the Softselect feature of the RSU-2 unit from Eramco, I set out to build a module that would greatly increase the easy of reviving a RSU-2 unit. The result is this INITRSU module which has to be loaded into page 6(!) to work.

It comes with a set of functions that help the user to restore operation, clear parts or all of the RSU and load the operating system, all of this for all available banks and in a very convenient and save way. Last but not least, it also includes the original SETBANK, PORTON and PORTOFF functions from the RSU operating system. For the curious user, please also find the documented source code at the end of this document. Especially the documentation of the SETBANK instruction might be interesting to study.

Quick-Reference-Guide for functions of INITRSU

XROM 04		
01	INIT-RSU	no function, module header
02	PORTON()	the original function to turn on a port (-1 = 8, +3 = F)
03	PORTOFF(1)	the original function to turn off a port
04	SETBANK(1)	the original function to select a bank from 1-4
05	CLPOLLA	function that clears all polling points in all pages and banks
06	CLRSUA	function that clears all pages in all banks. CAREFUL!
07	CLPOLLB	function that clears the polling points in bank specified via X-reg
08	CLRSUB	function that clears all pages in bank specified via X-reg
09	CLBL()	clear a block specified in X, can clear across page boundaries
10	COPYP(2)	copy page S to page D, SD in alpha
11	CODE(2)	code a NNN into X from Alpha
12	DECODE(2)	decode a NNN from X into Alpha
13	ROMSUM(2)	calculates and stores the correct checksum for a page
14	HEXIN()	direct entry of a NNN
15	BLCAT(3)	Block-cat showing the content of each page
16	RAMWR(2)	MCODE Hexeditor to write directly into the RSU
17	SAVEROM(2)	Save a page from X to HP-IL with name from alpha
18	GETROM(2)	Get a page from HP-IL with name in alpha to page in X
19	LOCA(2)	Find a specific MCODE word starting at address AAAA in alpha
20	LROM(2)	Find last word in MLDL starting search backwards from AAAA in alpha
21	LDRSU	Load the original operating system into pages E&F from HP-IL for all banks

Original functions come from the ERAMCO MLDL OS ,the Sandbox or the original RSU OS, though I did make certain changes here and there to the original code to fit the particular needs of this module.

In theory one would only need the functions CLRSUA (which clears all pages in all banks) and LDRSU (which loads the OS from HP-IL). However I find it useful to have some other tools at my disposal when trying to resurrect a dead Rambox.

File locations

You can find the complete manual [here](#), and the actual files [here](#).

Please let me know of any errors and suggestions.

Cheers

Peter

Edited: 18 Feb 2009, 6:41 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

HP-71B Math RAM

Posted by [Marcus von Cube, Germany](#) on 16 Feb 2009, 8:06 a.m.

This is not a typo! ;)

I managed to get the HP-71B Math Pac image into a free RAM port of my real HP-71B. Here is what I did.

Necessary Gear

1. A physical HP-71B with 32kB RAM to spare,
2. An ISA PC with HP-IL card or a PIL-BOX by J. F. Garnier,
3. HP-IL enabled EMU71 with MATHROM.BIN configured in some port,
4. A HEX editor on your desktop machine.

Modification to MATHROM.BIN

The ROM module contains a protected file which will not be copied by the HP-71B's COPY command unless you change the flag beforehand. This is quite easy: Open MATHROM.BIN with a HEX editor and change the byte 03 at address 000E to 00. Voilà, the protection is gone!

In order to save precious RAM you should set the file size in the image to a value that does not include the many 0 nibbles at the end of the ROM: Put the values 50 D8 at address 0014 in the file (original value was D6 FF.)

Copy the LEX file to a virtual drive

In EMU71 (with physical HP-71B turned off):

```
COPY MATHROM:PORT(2) TO :HDRIVE1
CONTROL OFF
```

Install the LEX on physical HP-71B

On physical HP-71B:

```
[ON]
COPY MATHROM:HDRIVE1 TO :PORT(5.01)
SECURE MATHROM
[OFF]
[ON]
```

And now to test:

```
DIM A(5)
```

```
MAT INPUT A
MAT PRINT A
```

Explanations

My EMU71 configuration has MATHROM.BIN configured in port 2. I'm pretty sure that you can even leave out the source port specification in the COPY command and EMU71 will still find the file.

I've a CMT card reader port RAM module which installs ports 5.01 to 5.04. 5.01 was just free for the purpose. That's all. If you have claimed all port RAM as main memory, you should be able to copy MATHROM in main RAM but I haven't tested it.

Good Math!

Edited: 5 May 2012, 4:58 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

HP41 - How to mix and match UCODE and MCODE in a module

Posted by [PeterP](#) on 13 Feb 2009, 7:37 p.m.

Hi,

I had a few times the wish to combine normal usercode programs and MCODE functions in a custom module. The procedure below works yet I wonder if there is a more efficient way to accomplish this.

Building MCODE functions is most easily done via the SDK41 package from WF. It allows the use of a normal text-editor on a PC to write the code and comment it. Additionally it provides a rapid testing environment where the newly written code can be immediately checked for errors and tested (in single step mode if wanted). AFAIK there is no faster MCODE development environment around. However, sometimes the need arises to add a user-code program to the MCODE functions in the same module and there is no fast way to convert a user-code program into the correct MCODE bytes that have to be loaded that I know of. So I am using emu41 and then SDK41 or M2k to generate the disassembled code that can then be easily meshed with the MCODE to create one annotated text-file for the whole module.

Prepare EMU41

Create a EMU41.ini file which includes the WWOS as well as a standard 8k MLDL RAM that is provided with EMU41. My .ini files looks like this

```
;-----
; emu41.ini          emu41 init file

[MODULES]
; page size filename
0  12 rom41cx.bin
3   4 cxfcn2d.bin
5   8 cxtime2c.bin
7   4 hpil1h.bin
;-----
8   4 WWOSa.bin
10  8 mldlram.dat
15  4 CCD_osx.bin

[DEVICES]
DISPLAY
HDRIVE1
;-----
```

It has the W&W Rambox OS on page 8, the 2 pages of free MLDL Ram on pages A&B and the CCD-OSX for convenience on page 15 (Thank you Raymond...)

Save your .ini file in the same directory as you have your emu41.exe file and run emu41.exe.

Enter your PGM

Now you can enter your program that you want as part of your MCODE module. If it is just a short module, I suggest to just type it in with emu41. If it is longer and you do have a HP-IL card you can also load it in from another PC via the HDRIVE1. (In short: Load the program into a physical HP via finger, Wand, cards, etc – save it to the HDRIVE1 via the HP-IL card – turn off the physical HP41 – load it into memory from HDRIVE1 with emu41).

Store the program in the MLDL Ram

Using the functions of the W&W OS we can easily prepare a page and store the program on it. The steps below assume you have the same emu41.ini file as above and the MLDL ram is at page 10 & 11 (A&B)

1. 10, xeq CLPG -----> this clears page 10
2. alpha UCODEMOD alpha -----> the name of the module
3. 8,enter,10 xew INITPG -----> this initializes the module with XROM number 08
4. alpha NAME_OF_PGM alpha
5. 10; xeq LDPGM -----> this compiles and stores your program in the MLDL
6. alpha UCODEMOD,RE-01 alpha ---> to add revision number and chksum at end of module
7. 10 xeq ENDPG, xeq PGSUM, xeq PGSUM, xeq ENDPG -----> this ensures that we have a valid module just in case

Convert MLDL Ram to .ROM

1. exit emu41
2. copy MLDLram.dat to MLDLRAM.bin
3. clbin41 MLDLRam.bin MLDLRAM.rom /b ;this converts the .bin file into a .rom file

Disassemble the .ROM file

There are two ways to disassemble the .rom file to get simple text that you can include into your other MCODE source text.

1. Use D41 from the SDK41 to get a .src file which has the exact hex codes needed to have the UCODE program in the MLDL-ram. The first two bytes are the header for the UCODE followed directly by the bytes for the global Label of your program.
2. Use M2K Manager's disassembly function to also generate a text file which has additionally the UCODE steps in it as well. However one needs to do some post-processing to use this file as an input in the SDK41. Alpha-strings and Alpha labels show their hex-codes to the right in the listing. This needs to be corrected by making them separate lines.

Personally I like the UCODE description from M2K and so I post-process them using the D41 assembly as a guideline. This gives me a nice code like the example below.

```
;-----
06BD 013                                #013                ;UserCode: 137 bytes (19 regs + 4 bytes)
NONPRIVATE
06BE 240                                #240
06BF 1CC                                #1CC                ;*LBL  "LDRSU"                ; 4C 44 52 53 55
06C0 000                                #000
06C1 0F6                                #0F6
06C2 000                                #000
06C3 04C                                #04C
06C4 044                                #044
06C5 052                                #052
06C6 053                                #053
06C7 055                                #055
```

```
06C8 1F9          #1F9          ;... 002  "RSU1 - FL : ?"      ;
06C9 052          #52
06CA 053          #53
06CB 055          #55
06CC 031          #31
06CD 02D          #2D
06CE 046          #46
06CF 04C          #4C
06D0 03A          #3A
06D1 03F          #3F
06D2 18C          #18C          ;... 003  AON
06D3 18E          #18E          ;... 004  PROMPT
06D4 18B          #18B          ;... 005  AOFF
06D5 111          #111          ;... 006  1
06D6 1A8          #1A8
06D7 000          #000          ;... 007  SF    00
06D8 1A1          #1A1
06D9 003          #003          ;... 008  XROM 04,03 - SETBANK
06DA 111          #111          ;... 009  14
06DB 1A1          #1A1
06DC 011          #011          ;... 010  XROM 04,17 - GETROM
;-----
```

This code can now easily be combined with the rest of your MCODE source file that you can compile with A41 & L41. Just remember that with the M2K Manager you can easily convert .ROM files into .MOD files in case you need them (for your iPhone for example or V41)

Please let me know of any improvements to this process.

Cheers

Peter

Edited: 17 Feb 2009, 12:29 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

cheaper batteries for the HP41CX

Posted by [Michael Harwood](#) on 30 Jan 2009, 9:32 p.m.

I found that the J size battery is a flat 6V battery and will fit into your HP-41CX. It is only \$3.00 (in Canada) and a lot easier to find. It took a bit of fiddling with wires to get the + and - to the HP contacts, but now it works. I can upload a photo if anyone is interested to see how I did it. * Note: I had an extra battery holder which I used after I removed the dividing sections for the N batteries.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

HP9815 - repairing the paper advance mechanism

Posted by [Mark Edmonds](#) on 19 Oct 2008, 8:29 a.m.

If your HP9815's printer is not advancing paper correctly, this may be a solution.

Two ways to identify the fault that is fixed here.

1. Does the paper advance slowly, intermittently and by very small amounts (one pixel line)?
2. When you print something, does the printer make a single "click" noise instead of multiple "click" noises?

If yes to either of those questions, read on.

Basically, what has probably happened is a foam pad has decayed inside the printer mechanism and left a horrible tar-like goo over part of the mechanism.

Step 1 is to open the machine.

1. Unplug and remove the fuse holder inside the paper roll bay.
2. Turn the machine over so the underside is facing up with the keyboard at the front.
3. Remove the top two screws. These are the top cover screws.
4. Remove the four screws in a large central rectangle underneath the keyboard area. These are the keyboard screws. The four screws in the smaller rectangle on the left hand side are the printer screws. Leave these for now.
5. Hold everything together and turn the machine over. Lift up the keyboard section a little and remove the rear top cover.
6. Now remove all connectors attached to the main keyboard assembly. There is one underneath that should be done first otherwise it might get pulled up unevenly and damage the pins. Put the keyboard assembly to one side.
7. Disconnect all leads to the printer assembly - 4 connectors then unscrew the printer assembly from the under side. Lift out the printer.
8. Hold the printer assembly so you are looking down the platen axis with the manual advance wheel nearest. There are three screws on the end panel that must be removed to disassemble the printer. Once undone, lift the end panel off. Some parts of the printer are under slight tension so it will expand out. Hold your hand round it to maintain its shape.
9. Now you can disassemble the printer which is easy. Just lift bits out and make sure you note the positions for re-assembly later.
10. There are two areas that need cleaning. The residue left by the decayed foam is really nasty. It is a sticky tar that is

difficult to clean and wont wash off your hands easily. Try to avoid touching it.

11. The first area to clean is on the magnet or metal bar attached to the stator. The area where the platen advance ratchet lever contacts the metal bar should be black and sticky.

12. To clean, I used a Stanley knife blade held at a narrow angle (like cleaning glass) and scraped the goo off. Once you have got most of it off, use a spirit rag over the bar to clean the last bits off.

13. The second area to clean is the platen ratchet advance lever. This is a metal pad which gets pulled to the magnet then released where upon a spring pushes it out and it engages with a notch in the advance wheel that pushes the platen round a small amount. The metal pad on the advance lever will be covered in the same black tar and because this is sticking to the magnet bar, the mecahnism never releases properly and the paper doesn't advance. Clean using the same technique as before. You will also find four mounting holes that need cleaning out. I used a toothpick for these.

14. Once you have got everything free of the tar residue, reassembly the entire machine, switch on and test.

15. What about replacing the foam pad? I didn't do this for two reasons. Firstly, I have no idea how big it should be and how dense the foam is. Replace it with something too hard and the lever will not be able to run its full travel. Also, the only purpose of the foam that I can see is to reduce noise slightly. As the printer is noisy anyway, I don't think there will be much difference. If I ever find out exactly what sort of material and what size is needed, I'll update this post.

15. If all is OK, the printer will now advance paper correctly and instead of a single click when you print, you will get a rapid-fire series of clicks, once for each line of pixels.

Good luck!

Mark

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Many Digits of Pi (HP32sii, HP-16c, HP-12c, HP-12cp, HP-19c, HP-30b, HP-15c)

Posted by [Katie Wasserman](#) on 17 Oct 2008, 11:56 p.m.

Here are several fairly simple programs for calculating many digits of pi on various HP calculators. My goal in writing these is to see how many digits of pi can be calculated and saved in memory on calculators of limited memory (under 1K bytes).

200 decimal digits of pi on the HP-32SII

XEQ P - results are in (A) - (Z) 8 digits per register no leading zeros - takes about 40 minutes.

based on:

$$\pi = 2 \times \sigma[i, 1, 1, \log_2(10) \cdot N, i/(2 \cdot i + 1)]$$
$$= 2 + \frac{1}{3} * (2 + \frac{2}{5} * (2 + \frac{3}{7} * (2 + \frac{4}{9} * (2 + \dots)))$$

where N = number of decimal digits

```
P01 | LBL P
P02 | CLVARs
P03 | CLSIGMA
```

```
P04 | 28          ; Set up to store the iteration counter in register n
P05 | STO i
```

```
P06 | 26          ; calculate number of iterations needed = log-base-2(10) * N
P07 | 8
P08 | x
P09 | 2
P10 | LOG
P11 | /
P12 | IP
P13 | STO(i)      ; Store in n
```

```
P14 | 2
P15 | STO A       ; starting value
P16 | x
P17 | 3
P18 | +
P19 | SIGMA+      ; the SIGMAx register is used to store (iteration_counter x 2 + 1)
```

```
B01 | LBL B       ; main loop
B02 | 1.026        ; set up for register loop
B03 | STO i
B04 | 0            ; initial carry
```

```
C01 | LBL C
```

```

C02 | 1E8      ; 100,000,000 this determines the number of digits per register
C03 | x

C04 | n
C05 | RCLx(i)
C06 | +      ; multiply and add carry

C07 | ENTER
C08 | ENTER
C09 | SIGMAx
C10 | /
C11 | IP      ; divide by (iteration_counter x 2 + 1)
C12 | STO(i)

C13 | SIGMAx
C14 | x
C15 | -      ; remainder is the carry

C16 | ISG i
C17 | GT0 C

C18 | 2
C19 | STO+ A
C20 | SIGMA- ; decrement iteration_counter by 1; and (2 x iteration_counter + 1) by 2
C21 | x>0?
C22 | GT0 B

C23 | 26      ; set up for register loop to carry overflows up one register,
C24 | STO i   ; register value can never be more than 2 x 10^8

E01 | LBL E
E02 | RCL(i)
E03 | 1E8
E04 | -
E05 | x<=0?
E06 | GT0 F   ; no carry
E07 | STO(i)
E08 | 1
E09 | STO- i   ; these 3 instructions add the carry forward
E10 | STO+(i)
E11 | STO+ i
E12 | GT0 E   ; this isn't really needed since the overflow can never be more than 1E8,
               ; but it's nice to have if other algorithms are used

F01 | LBL F
F02 | DSE i
F03 | GT0 E
F04 | RTN

```

Checksums:

```

P=28.5 CK=76B9
B=14.0 CK=334E
C=44.0 CK=DFE2
E=26.0 CK=C6AE
F=06.0 CK=BF13

```

195 decimal digits of pi on the HP-16C

GSB A - results are in (0) - (F) 13 digits per register no leading zeros - takes about 17 hours.

based on:

$$\begin{aligned}
 \pi &= 2 \times \text{sigma}[i, 1, 1, \log_2(10)^N, i/(2^i+1)] \\
 &= 2 + 1/3 * (2 + 2/5 * (2 + 3/7 * (2 + 4/9 * (2 + \dots
 \end{aligned}$$

where N = number of decimal digits

```

001 | LBL A | 43,22, A

002 | FLOAT 0 | 42,45, 0 ; Set 56 bit word size, 2's compliment
003 | DEC | 24
004 | CLR REG | 42 34
005 | 2S COMP | 42 2

006 | 6 | 6 ; 663 iterations = 195 digits, convergence is about 1 bit per
iteration
007 | 6 | 6
008 | 3 | 3

009 | LBL B | 43,22, B
010 | STO .0 | 44 .0 ; register .0 is the iteration_counter

011 | 0 | 0 ; initial carry

012 | STO I | 44 32 ; register I is the register pointer (0 - F)

013 | LBL C | 43,22, C
014 | GSB 8 | 21 8 ; get 10,000,000,000,000
015 | X | 20 ; Carry x 10^13

016 | RCL .0 | 45 .0
017 | RCL (i) | 45 31
018 | X | 20
019 | + | 40 ; multiply and add carry

020 | Enter | 36
021 | Enter | 36

022 | RCL .0 | 45 .0
023 | SL | 42 A
024 | 1 | 1
025 | + | 40
026 | / | 10 ; divide by (iteration_counter x 2 + 1)

027 | STO (i) | 44 31 ; store result

028 | R down | 33
029 | LST X | 43 36
030 | RMD | 42 9 ; remainder is the carry

031 | X <> I | 42 22 ; increment register pointer and check for end of loop
032 | 1 | 1
033 | 5 | 5
034 | X = Y | 43 49
035 | GTO D | 22 D
036 | R down | 33
037 | 1 | 1
038 | + | 40
039 | X <> I | 42 22
040 | GTO C | 22 C

041 | LBL D | 43,22, D ; decrement iteration_counter and check for end
042 | 2 | 2
043 | STO 0 | 44 0
044 | RCL .0 | 45 .0
045 | 1 | 1
046 | - | 30
047 | X > 0 | 43 30
048 | GTO B | 22 B

```

```
049      1      1      ; loop through registers to carry any overflows up to
050      5      5      ; next register. Overflow can never be more than 2 x 10^13.
051      STO I      44 32
052      0      0
053      LBL E      43,22, E
054      GSB 7      21 7      ; this is weird, but saves one byte
055      GSB 8      21 8
056      -      30
057      X < 0      43 2
058      GT0 F      22 F
059      STO (i)     44 31
060      1      1
061      GT0 9      22 9
062      LBL F      43,22, F
063      0      0
064      LBL 9      43,22, 9
065      DSZ      43 23
066      GT0 E      22 E
067      LBL 7      43,22, 7
068      RCL (i)     45 31
069      +      40
070      STO (i)     44 31
071      RTN      43 21
```

```
072      LBL 8      43,22, 8 ; put 10^13 on the stack
073      1      1
074      0      0
075      0      0
076      0      0
077      Enter      36
078      X      20
079      Enter      36
080      X      20
081      1      1
082      0      0
083      X      20
084      RTN      43 21
```

70 decimal digits of pi on the original HP-12C

CLEAR REG
R/S - results are in (0) - (.0) 7 digits per register no leading zeros - takes about 90 minutes.

based on:

$$\pi = 2 \times \sigma[i,1,1,\log_2(10)^N,i/(2^i+1)]$$
$$= 2 + \frac{1}{3} * (2 + \frac{2}{5} * (2 + \frac{3}{7} * (2 + \frac{4}{9} * (2 + \dots)))$$

where N = number of decimal digits

```
01      EEX      26
02      7      7
03      STO FV      44 15
04      2      2
05      4      4
06      0      0
07      STO i      44 12
08      GT0 42      43,33 42
09      0      0
10      STO n      44 11
11      STO PMT      44 14
12      RCL PMT      45 14
13      RCL FV      45 15
14      x      20
15      RCL CFj      45,43 14
16      RCL i      45 12
17      x      20
18      +      40
19      ENTER      36
20      ENTER      36
21      RCL i      45 12
22      2      2
23      x      20
```


24	1		1
25	+		40
26	STO PV	44	13
27	/		10
28	INTG	43	25
29	CFj	43	14
30	RCL PV	45	13
31	x		20
32	-		30
33	STO PMT	44	14
34	1		1
35	0		0
36	RCL n	45	11
37	1		1
38	+		40
39	STO n	44	11
40	x<=y	43	34
41	GTO 12	43,33	12
42	2		2
43	STO 0	44	0
44	RCL i	45	12
45	1		1
46	-		30
47	STO i	44	12
48	x=0	43	35
49	GTO 51	43,33	51
50	GTO 09	43,33	09
51	1		1
52	0		0
53	STO n	44	11
54	RCL i	45	12
55	RCL CFj	45,43	14
56	+		40
57	RCL FV	45	15
58	/		10
59	INTG	43	25
60	STO i	44	12
61	LASTx	43	36
62	FRAC	43	24
63	RCL FV	45	15
64	x		20
65	CFj	43	14
66	RCL CFj	45,43	14
67	RCL n	45	11
68	0		0
69	x<=y	43	34
70	GTO 54	43,33	54
71	GTO 00	43,33	00

632 decimal digits of pi on the HP-12C Platinum (w/80 CF registers)

R/S - results are in (0) - (79) 8 digits per register no leading zeros - takes about 45 hours.

This program will show register (0) when it ends and start a continuous loop thru all the registers, advanced each time R/S is pressed. IMPORTANT NOTE: If the calculator times-out and shuts down before you get a chance to press R/S you must do a .GTO 081 after turning the calculator back on. Failure to do this will cause the program to re-run wiping out your result. This happens because the calculator resets the program pointer to 000 when it turns on.

based on:

$$\pi = 2 \times \text{sigma}[i,1,1,\log_2(10)*N,i/(2*i+1)]$$
$$= 2 + 1/3 * (2 + 2/5 * (2 + 3/7 * (2 + 4/9 * (2 + \dots$$

where N = number of decimal digits

001	RPN	42	16
002	CLEAR FIN	42	34
003	0		0
004	CFj	43	14

005	7	7
006	9	9
007	RCL n	45 11
008	x<=y	43 34
009	GT0 003	43,33,003
010	(F) 0	42 0
011	2	2
012	1	1
013	0	0
014	0	0
015	ST0 i	44 12
016	EEX	26
017	8	8
018	ST0 FV	44 15
019	0	0
020	ST0 n	44 11
021	ST0 PMT	44 14
022	RCL PMT	45 14
023	RCL FV	45 15
024	x	20
025	RCL CFj	45,43 14
026	RCL i	45 12
027	x	20
028	+	40
029	ENTER	36
030	ENTER	36
031	RCL i	45 12
032	2	2
033	x	20
034	1	1
035	+	40
036	ST0 PV	44 13
037	/	10
038	INTG	43 25
039	CFj	43 14
040	RCL PV	45 13
041	x	20
042	-	30
043	ST0 PMT	44 14
044	7	7
045	9	9
046	RCL n	45 11
047	1	1
048	+	40
049	ST0 n	44 11
050	x<=y	43 34
051	GT0 022	43,33,022
052	2	2
053	ST0 0	44 0
054	RCL i	45 12
055	1	1
056	-	30
057	ST0 i	44 12
058	x=0	43 35
059	GT0 061	43,33,061
060	GT0 019	43,33,019
061	7	7
062	9	9
063	ST0 n	44 11
064	RCL i	45 12
065	RCL CFj	45,43 14
066	+	40
067	RCL FV	45 15
068	/	10
069	INTG	43 25
070	ST0 i	44 12
071	LASTx	43 40
072	FRAC	43 24
073	RCL FV	45 15
074	x	20
075	CFj	43 14
076	RCL CFj	45,43 14
077	RCL n	45 11
078	0	0
079	x<=y	43 34
080	GT0 064	43,33,064
081	0	0
082	ST0 n	44 11
083	RCL CFj	45,43 14
084	R/S	31
085	7	7

086	9	9
087	RCL n	45 11
088	2	2
089	+	40
090	x<=y	43 34
091	GTO 082	43,33,082
092	GTO 081	43,33,081

162 decimal digits of pi on the HP-19C

GSB 0 - results are printed are in (2) - (29), 6 digits per register no leading zeros
 - takes about 8.5 hours. Also, a progress countdown is given showing the iteration counter.
 After a few minutes you'll see 53 print than 52, 51, ... every 9.5 minutes or so.

based on:

$$\pi = 2 \times \sigma[i,1,1,\log_2(10)*N,i/(2*i+1)]$$

$$= 2 + 1/3 * (2 + 2/5 * (2 + 3/7 * (2 + 4/9 * (2 + \dots$$

where N = number of decimal digits

01	*LBL 0	25 14 00
02	FIX0	16 13 00
03	CLRG	16 23
04	5	05
05	3	03
06	8	08 ; 538 iterations = 162 digits, convergence is about 1 bit per
iteration		
07	ST01	45 01

08	*LBL1	25 14 01 ; top of main loop

09	RCL1	55 01
10	1	01
11	0	00
12	/	61
13	INT	16 52
14	LSTX	16 63
15	X=Y?	16 61
16	PRTX	65 ; print status every 10 iterations

17	2	02
18	ST00	45 00
19	0	00 ; initial value of carry

20	*LBL2	25 14 02 ; top of register loop
21	EEX	23
22	6	06
23	x	51
24	RCLi	55 12
25	RCL1	55 01
26	x	51
27	+	41
28	ENT^	21
29	ENT^	21
30	ENT^	21
31	RCL1	55 01
32	2	02
33	x	51
34	1	01
35	+	41
36	/	61
37	LSTX	16 63
38	X<>Y	11
39	INT	16 52
40	ST0i	45 12
41	x	51
42	-	31
43	ISZ	25 55
44	RCL0	55 00
45	3	03

46		0		00	
47		-		31	
48		X=0?		25	61
49		GT03		14	03
50		Rv			12
51		GT02		14	02
52		*LBL3		25	14 03
53		2			02
54		ST02		45	02
55		RCL1		55	01
56		1			01
57		-			31
58		ST01		45	01
59		X>0?		25	41
60		GT01		14	01
61		2		02	; loop through registers and carry overflow up
62		9		09	
63		ST00		45	00
64		0			00
65		*LBL4		25	14 04
66		ST+i		45	41 12
67		RCLi		55	12
68		EEX			23
69		6			06
70		/			61
71		INT		16	52
72		ST01		45	01
73		EEX			23
74		6			06
75		x			51
76		ST-i		45	31 12
77		RCL1		55	01
78		DSZ		25	45
79		GT04		14	04
80		2		02	; print results in registers (2) through (29)
81		ST00		45	00
82		*LBL5		25	14 05
83		RCLi		55	12
84		PRTX			65
85		1			01
86		ST+0		45	41 00
87		RCL0		55	00
88		3			03
89		0			00
90		X>Y?		16	41
91		GOT5		14	05
92		RTN		25	13
93		R/S			64

693 decimal digits of pi on the HP-30b

Put 99 (maximum number, you can use less) on the stack and run.

- results are in Data(X) and Data(Y), 7 digits per register
no leading zeros
- takes about 39 minutes (no other programs in the calc)
- a progress countdown is given showing the iteration counter

based on:

$$pi = 2 \times \sigma[i, 1, 1, \log_2(10) \cdot N, i / (2^i + 1)]$$

$$= 2 + \frac{1}{3} * (2 + \frac{2}{5} * (2 + \frac{3}{7} * (2 + \frac{4}{9} * (2 + \dots)))$$

where N = number of decimal digits

```

1 STO 9 -- Save the number of registers to fill

2 RCL Mode -- save the Mode
3 STO 5
4 3
5 0
6 0
7 STO Mode -- Set RPN and FIX 00

8 Reset -- Clear the CF registers to make maximum room in the Stats Registers
9 DOWN
10 DOWN
11 INPUT
12 INPUT
13 Reset
14 DOWN
15 DOWN
16 DOWN
17 INPUT
18 INPUT

19 RCL 9 -- Figure out how many iterations will be needed based on the number of registers to
fill
20 7 -- 7 digits per register
21 *
22 2
23 *
24 2
25 LN
26 1
27 0
28 LN
29 /
30 / - # digits x log-base-2(10) are the iterations needed (and we count down by 2)

31 MATH -- take the IP
32 UP
33 UP
34 INPUT
35 STO 1 -- and save in 1

36 LBL 80 - top of main loop
37 RCL 9
38 EEX
39 3
40 /
41 STO 0 -- set up number of register loop for ISG
42 2 -- initial value of term
43 STO Data
44 0 -- initial carry
45 INPUT

46 LBL 81 -- top of loop through the Stat registers
47 EEX
48 7
49 *
50 RCL DATA
51 RCL x 1
52 + -- numerator is Data(i)*n + carry * 10^7
53 STO 2
54 RCL 1
55 Disp0
56 2
57 *
58 1
59 + -- denominator is 2n+1
60 STO 3
61 /
62 MATH -- take the IP
63 UP
64 UP
65 INPUT
66 STO Data -- IP((Data(i)*n+carry*10^7)/(2n+1)) --> Data(i)

```

```

67 RCL x 3
68 RCL 2
69 SWAP
70 -          -- carry into next register
71 ISG 0      -- check for end of Stat register loop
72 GTO 81

73 0          -- next 4 lines are needed because of the way the loop ends, we need "2" in
Data(0)
74 STO 0
75 2
76 STO Data
77 DSE 1      -- check for end of term loop
78 GTO 80

79 RCL 9      -- This is the final "fix up" loop, just one pass through the registers to adjust the
overflows
80 STO 0
81 0          -- initial value of carry
82 INPUT
83 LBL 82
84 RCL Data
85 +
86 STO Data  -- Add carry to register
87 EEX
88 7
89 ?<
90 GT 83      -- no overflow so skip over adjustment
91 RCL Data
92 EEX
93 7
94 -          -- back out overflow
95 STO Data
96 1          -- carry into next register
97 INPUT
98 GTO 84
99 LBL 83
100 0         -- no carry
101 INPUT
102 LBL 84
103 DSE 0      -- decrement register pointer and loop
104 GTO 82
105 RCL Data  -- add carry to Data(0)
106 +
107 STO Data

108 RCL 5      -- done, restore mode settings
109 STO Mode
110 Data      -- leave the user in Data review mode to see the results
111 Stop

```

```

-----
157 Bytes
014 Checksum

```

306 decimal digits of pi on the HP-15C

GSB 0 - results in (2) - (53), 6 digits per register no leading zeros.
 It takes about 43 hours to run on an original 15C or 16.1 minutes on a 15C LE.

based on:

$$\pi = 2 \times \sigma[i, 1, 1, \log_2(10) \cdot N, i / (2 \cdot i + 1)] \\
 = 2 + \frac{1}{3} * (2 + \frac{2}{5} * (2 + \frac{3}{7} * (2 + \frac{4}{9} * (2 + \dots)))$$

where N = number of decimal digits

01	*LBL 0	42, 21, 0
02	5	5
03	3	3

04	DIM (i)	42, 23, 24
05	FIX 0	42, 7, 0
06	CLR REG	42 34
07	1	1
08	0	0
09	1	1
10	7	7 ; 1017 iterations = 306 digits, convergence is about 1 bit per iteration
11	STO 1	44 1

12	*LBL 1	42, 21, 1 ; top of main loop
13	2	2
14	STO I	44 25
15	0	0 ; initial value of carry

16	*LBL 2	42, 21, 2 ; top of register loop
17	EEX	26
18	6	6
19	x	20
20	RCL (i)	45 24
21	RCL 1	45 1
22	x	20
23	+	40
24	ENTER	36
25	ENTER	36
26	ENTER	36
27	RCL 1	45 1
28	2	2
29	x	20
30	1	1
31	+	40
32	/	10
33	LST x	43 36
34	X<>Y	34
35	INT	43 44
36	STO (i)	44 24
37	x	20
38	-	30
39	1	1
40	STO+I	44, 40, 25
41	RV	33
42	RCL I	45 25
43	5	5
44	4	4
45	-	30
46	X=0?	43 20
47	GTO 3	22 3
48	RV	33
49	GTO 2	22 2

50	*LBL 3	42, 21, 3
51	2	2
52	STO 2	44 2
53	RCL 1	45 1
54	1	1
55	-	30
56	STO 1	44 1
57	X>0?	43, 30, 1 (TEST 1)
58	GTO 1	22 1

59	5	5 ; loop through registers and carry overflow up
60	3	3
61	STO I	44 25
62	0	0

63	*LBL 4	42, 21, 4
64	STO+(i)	44, 40, 24
65	RCL (i)	45 24
66	EEX	26
67	6	6
68	/	10
69	INT	43 44
70	STO 1	44 1
71	EEX	26
72	6	6

73		x		20
74		STO-(i)		44, 30, 24
75		RCL 1		45 1
76		DSE I		42, 5, 25
77		GTO 4		22 4
78		RCL 2		45 2
79		RTN		43 32

Edited: 20 Jan 2012, 3:18 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP67/97 Navigation Pac I - Star Sight Planner Data Card Update for 2008

Posted by [MikeO](#) on 11 Aug 2008, 3:45 p.m.

For those interested in HP67/97 Nav Pack I:

A friend of mine, who's using my [HP97 emulator](#), asked me if I could look at the Navpac Star Sight Planner program (NAV-05A1), and figure out how to update the star data on the data card (NAV-05A2). Using the emulator, I reverse engineered the data extraction formulas used by the program, and used them to figure out what was wrong with the compression formula given in the manual.

Using current nautical almanac data, I was then able to produce a new version of the NAV-05A2 data card that results in about a 22% improvement in accuracy. The data card contents are listed below:

Encoded Star Data
As of 8/8/2008
Replaces NAV-05A2 Data Card

0.0000000	0	-	
0.3393721	1		
18386.60088	2		
81716.37301	3		
72263.47855	4		
22270.40621	5	-	Primary Registers
54475.24512	6		
46255.79618	7		
59588.47426	8		
34622.74119	9	-	
89094.55238	0	-	
43536.15781	1		
41856.60219	2		
85191.19147	3		
56337.23035	4		
39743.84219	5	-	Secondary Registers
9228.353652	6		
64772.70976	7		
43174.28825	8		
64906.45485	9	-	
82117.25485	A		
44888.67165	B		
0.0000000	C		
0.0000000	D		
0.0000000	E		

Data Reference:

<http://www.tecepe.com.br/scripts/AlmanacPagesISAPI.isa/pages?date=08%2F08%2F2008>

I hope this comes in handy!

-MikeO

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-35S Multiple cashflow / NPV / IRR / NFV / NUS program

Posted by [Caspar L](#) on 1 Aug 2008, 2:18 p.m.

Hello all,

Please find below my financial program for the HP-35s calculator for multiple cashflows / investment analysis.

Note:

- indirect registers are used for the multiple cashflows
- indirect registers are NOT cleared after end of program
- when run again previous values of the cashflows are recalled to stack level T (ready to be used again), the cashflow counter is shown on stack level X
- after entering each cashflow value press the [R/S] key to continue
- the number of cashflows is limited by memory only
- two labels are used because the solver is used with a program to calculate the IRR (label R program)
- obviously you can use different labels (instead of C and R)
- to continue after solution for P etc is shown just press [R/S] key

```

C001      LBL C
C002      SF10
C003      INPUT N
C004      1
C005      +
C006      STO J
C007      1
C008      STO (J)
C009      INPUT I
C010      STO R
C011      0
C012      STO J
C013      EQN CF0,CF1, ,CFN
C014      PSE
C015      RCL J
C016      RCL (J)
C017      STOP
C018      STO (J)
C019      1
C020      STO+ J
C021      RCL J
C022      RCL N
C023      x<y?
C024      GT0 C027
C025      Rdown
C026      GT0 C015
C027      0
C028      STO P
C029      0
C030      STO J
C031      RCL (J)
C032      1
C033      RCL I
C034      100
C035      /
C036      +
C037      RCL J
C038      ^
C039      /
C040      STO+ P
    
```

```
C041      RCL J
C042      RCL N
C043      x=y?
C044      GT0 C048
C045      1
C046      ST0+ J
C047      GT0 C031
C048      VIEW P
C049      0
C050      ST0 F
C051      0
C052      ST0 J
C053      RCL (J)
C054      1
C055      RCL I
C056      100
C057      /
C058      +
C059      RCL N
C060      RCL J
C061      -
C062      ^
C063      *
C064      ST0+ F
C065      RCL J
C066      RCL N
C067      x=y?
C068      GT0 C072
C069      1
C070      ST0+ J
C071      GT0 C053
C072      VIEW F
C073      CF 10
C074      EQN P/((1-(1/(1+I/100)^N))/(I/100))
C075      ST0 U
C076      VIEW U
C077      0
C078      ST0 Q
C079      0
C080      ST0 J
C081      FN= R
C082      SOLVE R
C083      VIEW R
C084      RCL P
C085      RCL F
C086      RCL U
C087      RCL R
C088      RTN
```

```
R001      LBL R
R002      RCL (J)
R003      1
R004      RCL R
R005      100
R006      /
R007      +
R008      RCL J
R009      ^
R010      /
R011      ST0+ Q
R012      RCL J
R013      RCL N
R014      x=y?
R015      GT0 R019
R016      1
R017      ST0+ J
R018      GT0 R002
R019      RCL Q
R020      0
R021      ST0 J
R022      0
R023      ST0 Q
R024      Rdown
R025      Rdown
R026      RTN
```

variables used:
N = number of periods, excluding investment at t=0

I = interest or cost of capital in %
J = loop counter for cashflows
P = NPV
F = NFV
U = NUS
R = IRR in %
Q = temporary variable for solver

An example:

```
*INPUT*
N=8
I=10
CF0=-130000
CF1=7000
CF2=-10000
CF3=20000
CF4=20000
CF5=20000
CF6=12000
CF7=-8000
CF8=178500

*OUTPUT*
NPV=-4855.85
NFV=-10408.94
NUS=-910.20
IRR=9.37%
```

The output is available in the aforementioned registers and on the 4 stack levels.
Note that $NPV < 0$ (and $IRR < I$), thus the project is not worth undertaking.

Have fun with the program. In case of questions or suggestions, please send an e-mail.

Caspar

Edited: 24 Aug 2008, 5:09 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

HP-35S Triangle solve program

Posted by [Caspar L](#) on 24 July 2008, 4:36 p.m.

My first go at a 35S program (see FEM48/49 at hpcalc.org for a more elaborate program of mine) and my first posting here.

```
T001      LBL T
T002      CLX
T003      SF10
T004      EQN 1SSS 2SAS 3ASA4SAA 5SSA 6SSA
T005      x=0?
T006      GT0 T108
T007      1
T008      x=y?
T009      GT0 T032
T010      R Down
T011      2
T012      x=y?
T013      GT0 T039
T014      R Down
T015      3
T016      x=y?
T017      GT0 T046
T018      R Down
T019      4
T020      x=y?
T021      GT0 T053
T022      R Down
T023      5
T024      x=y?
T025      GT0 T060
T026      R Down
T027      6
T028      x=y?
T029      GT0 T062
T030      EQN INVALID CMD
T031      GT0 T001
T032      EQN ENTER S1
T033      STO R
T034      EQN ENTER S2
T035      STO S
T036      EQN ENTER S3
T037      STO T
T038      GT0 T070
T039      EQN ENTER S1
T040      STO R
T041      EQN ENTER A3
T042      STO W
T043      EQN ENTER S2
T044      STO S
T045      GT0 T080
T046      EQN ENTER A2
T047      STO V
T048      EQN ENTER S1
T049      STO R
T050      EQN ENTER A3
T051      STO W
T052      GT0 T084
T053      EQN ENTER S1
T054      STO R
T055      EQN ENTER A3
```



```
T056      ST0 W
T057      EQN ENTER A1
T058      ST0 U
T059      GT0 T092
T060      SF 1
T061      GT0 T063
T062      CF 1
T063      EQN ENTER S1
T064      ST0 R
T065      EQN ENTER S2
T066      ST0 S
T067      EQN ENTER A1
T068      ST0 U
T069      GT0 T096
T070      CF 10
T071      EQN (R+S+T)/2
T072      ST0 Q
T073      EQN 2*ACOS((Q*(Q-R)/(S*T))^0.5)
T074      ST0 U
T075      EQN 2*ACOS((Q*(Q-S)/(R*T))^0.5)
T076      ST0 V
T077      EQN ACOS( -COS(U+V))
T078      ST0 W
T079      GT0 T106
T080      CF 10
T081      EQN (R*R+S*S-2*R*S*COS(W))^0.5
T082      ST0 T
T083      GT0 T070
T084      CF 10
T085      EQN ACOS( -COS(V+W))
T086      ST0 U
T087      EQN R*(SIN(V)/SIN(U))
T088      ST0 S
T089      EQN R*COS(V)+S*COS(U)
T090      ST0 T
T091      GT0 T106
T092      CF 10
T093      EQN ACOS( -COS(U+W))
T094      ST0 V
T095      GT0 T084
T096      CF 10
T097      EQN ASIN(S/R*SIN(U))
T098      ST0 V
T099      EQN ACOS( -COS(U+V))
T100      ST0 W
T101      FS? 1
T102      GT0 T084
T103      EQN ACOS(1)-W
T104      ST0 W
T105      GT0 T084
T106      EQN R*S*SIN(W)/2
T107      ST0 Q
T108      CF 1
T109      CF 10
T110      RTN
```

Variables used:

area	Q
S1	R
S2	S
S3	T
A1	U
A2	V
A3	W

The program solves for 5 different triangle problems, where the SSA problem has two solutions:

- 1SSS
- 2SAS
- 3ASA
- 4SAA
- 5SSA
- 6SSA

The program solves for any plane triangle's angles, sides and area, when given the above three inputs.

After a prompt for an input (e.g. ENTER S1) you enter the numeric value and press the [R/S] key to continue.
Angles can be in deg, rad or grad, depending on your MODE.
I modelled the interface after a program of Stefan Vorkoetter which I liked a lot.
Please note that there is another program out there which does the same thing but needs 389 lines and uses 11 variables.
If so desired, you can enter the following two lines starting at T108 to enables quick viewing of the results, wihout recalling them from a register:
EQN [R,S,T]
EQN [U,V,W]
... (rest of lines follow)

Have fun, Caspar

Edited: 24 Aug 2008, 5:16 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

A Third Order Linear Equation Solver for the HP-35S

Posted by [Palmer O. Hanson, Jr.](#) on 20 May 2008, 11:24 p.m.

Program Description

This program is similar to my "A Label Saving Simultaneous Equation Solution for the 33s" (Article No. 599) but with modifications for use with the HP-35S. It will solve with real or complex coefficients. I wanted to use the same definition of the linear system and input routine as in the "Solution of Simultaneous Equations" program in pages 15-12 through 15-20 of the *hp 33s user's guide*. I couldn't because the indirect addressing registers (I and J) for the HP 35S are in the middle of the alphabetic data registers (a bad feature of the 35S to my way of thinking). Thus, the linear system is defined as

$$Ax + Dy + Gz = K$$

$$Bx + Ey + Hz = L$$

$$Cx + Fy + Iz = M$$

Where the definition of the matrix is the same but the definition of the vector uses data registers K through M rather than J through L as in the hp 33s manual..

The solution is obtained using Cramer's rule. The determinant subroutine is a copy of the subroutine on pages 15-16 and 15-17 of the *hp 33s user's guide*. The subroutine call is made using direct addressing to avoid using additional labels. The input data is restored at the completion of the solution. If the determinant is zero the calculator will stop with the message DIVIDE BY 0 in the display.

User Instructions

1. Press XEQ A ENTER to start the program.
2. The display will stop with a prompt for the coefficient to be entered. Press R/S to accept the displayed data or enter a new coefficient and press R/S. For complex coefficients use the Re i Im format.
3. The display will stop with a prompt for the next coefficient to be entered. Note that the entry is column by column not row by row.
4. Continue to enter the remaining coefficients in response to the prompts.
5. When you press R/S after entering the M value of the vector the program will proceed with the solution. If the determinant of the input matrix is zero program execution will stop with the message DIVIDE BY 0 in the display.
6. If the determinant of the input matrix is not zero the program will stop with value of X in the display. Press R/S to see the value of Y. Press R/S again to see the value of Z.

7. Press R/S to return to the start of the input routine.and see the first element of the input matrix.

8. The solution values can be recalled for further use from data registers X, Y and Z. The determinant can be recalled for further use from data register W.

Program Listing

```

A001 LBL A
A002 -1.1
A003 STO J
A004 INPUT (J)
A005 1.1
A006 STO- J
A007 -14.3
A008 RCL J
A009 x ne y?          x is not equal to y
A010 GTO A004
A011 XEQ A070
A012 STO W
A013 1/x
A014 RCL K
A015 x<> A
A016 STO K
A017 RCL L
A018 x<> B
A019 STO L
A020 RCL M
A021 x<> C
A022 STO M
A023 XEQ A070
A024 STO X
A025 RCL K
A026 x<> A
A027 x<> D
A028 STO K
A029 RCL L
A030 x<> B
A031 x<> E
A032 STO L
A033 RCL M
A034 x<> C
A035 x<> F
A036 STO M
A037 XEQ A070
A038 STO Y
A039 RCL K
A040 x<> D
A041 x<> G
A042 STO K
A043 RCL L
A044 x<> E
A045 x<> H
A046 STO L
A047 RCL M
A048 x<> F
A049 x<> I
A050 STO M
A051 XEQ A070
A052 STO Z
A053 RCL K
A054 x<> G
A055 STO K
A056 RCL L
A057 x<> H
A058 STO L
A059 RCL M
A060 x<> I
A061 STO M
A062 RCL W
A063 STO/ X
A064 STO/ Y
A065 STO/ Z
A066 VIEW X
A067 VIEW Y
A068 VIEW Z
A069 GTO A001
A070 RCL A
A071 RCLx E

```

```
A072 RCLx I
A073 RCL D
A074 RCLx H
A075 RCLx C
A076 +
A077 RCL G
A078 RCLx F
A079 RCLx B
A080 +
A081 RCL G
A082 RCLx E
A083 RCLx C
A084 -
A085 RCL A
A086 RCLx F
A087 RCLx H
A088 -
A089 RCL D
A090 RCLx B
A091 RCLx I
A092 -
A093 RTN
```

Length = 291; Check Sum = 2C23

Test problem:

The first number in each coefficient is the real part. The second number in each part is the imaginary part..

+1, +3	-1, +2	+4, 0	+10, -8
-3, -2	-5, +1	+1, +1	+10, -4
+2, 0	0, +3	-3, +4	+25, +12

The determinant of the matrix which can be read from W is +155,-66 .

The solution vector is

- +3, +1
- 2, -2
- +1, -4

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

An Extension for Stefan's Matrix Multi-Tool Program

Posted by [Palmer O. Hanson, Jr.](#) on 4 May 2008, 9:24 a.m.

In a thread starting on of November 2, 2007 Stefan Vorkoetter announced that a Matrix Multi-Tool program for the HP-35s was available at his site http://www.stefanv.com/calculators/hp35s_matrix_multitool.html . He noted that his program "... is a long way from being as powerful as the facilities provided by the 15C, 41C/Advantage, or 42s, but it does provide several useful matrix operations in one simple tool. It is loosely modelled after the Advantage Pac matrix *program* and provides an easy-to-use subset of the functionality of the matrix *library*. ..."

This program was written to work with Stefan's program and provide more of the matrix processing functions that one finds in machines such as the HP-28S and TI-86. This program relies on Stefan's program for entry and readout of data, and for solution of linear equations. Stefan's program uses one matrix (Matrix A) and one vector (Vector B) for input and output. His program also uses a second matrix (which I shall call Matrix C) during the linear equation solution. Matrix C is not directly accessible through Stefan's input and output routines. This program adds a storage matrix (which I shall call Matrix D). Additional matrix processing functions provided by this program include:

Calculation of Frobenius norm (ABS), the row norm, and the column norm of Matrix A.

Moving of matrices between Stefan's matrix A and either his matrix C or the added storage matrix D using store, recall and exchange commands. Moving of matrices directly between Matrix C and Matrix D is not provided.

Matrix mathematics including the product of Matrix A and Vector b, the product of Matrix A and Matrix C, and the sum or difference between Matrix A and Matrix C.

Special operations such as the dot product of Matrix A and Matrix C, the formation of an identity matrix in Matrix A, the filling Matrix A with a user-defined constant value, and the transpose of Matrix A..

This program uses a menu system similar to that used in Stefan's program. Like Stefan's program it saves labels by having all routines part of the main program label. At program entry and at the completion of each function the menu 1NRM 2MOV 3MTH 4OPS 5XXX appears in the display.

Selection the NRM option brings the menu 1ABS 2COL 3ROW to the display. Pressing any number key other than 1, 2, or 3 and pressing R/S returns the program to the entry menu.. Entering 1, 2, or 3 and pressing R/S brings the Frobenius norm (ABS), the column norm (COL) or the row norm (ROW) of contents of Matrix A to the display. Then pressing R/S returns the calculator to the entry menu. Matrix A is not disturbed by these operations.

Selection of th MOV option from the main menu brings the menu 1STOC 4STOD 2RCLC 5RCLD 3EXCC 6EXCD to the display. Note that there are two spaces between each option in the program listing. This avoids splitting the option nomenclature during display.

The STO commands copy the contents Matrix A into Matrix C or Matrix D.

The RCL commands copy the contents of Matrix C or Matrix D into Matrix A

The EXC commands exchange the contents of Matrix A with the contents of Matrix C or Matrix D.

Selection of the MTH option from the main menu brings the menu 1AxB 2AxC 3A+C 4A-C to the display.

The AxB command multiplies the contents of Matrix A by the contents of Vector B and places the result in Vector B and in the left hand column of Matrix C.

The AxC command multiplies the contents of Matrix A times the contents of Matrix C and places the result in Matrix A. The contents of Matrix A before the multiply are placed in Matrix C.

The A+C command adds the contents of Matrix A and Matrix C and places the result in Matrix A. The contents of Matrix A before the addition are placed in Matrix C.

The A-C command subtracts the contents of Matrix C from the contents of Matrix A and places the result in Matrix A. The contents of Matrix A before the subtraction are placed in Matrix C.

Selection of the OPS option from the main menu brings the menu 1DOT 2IDN 3FIL 4TRN to the display.

The DOT command calculates the dot product of the contents of Matrix A and Matrix C and delivers the result to the display.

The IDN command replaces the contents of Matrix A with an identity matrix.

The FIL command replaces the contents of Matrix A with a matrix which has all elements equal to a value selected by the user.

The TRN command replaces the contents of Matrix A with its transpose and leaves the matrix which was formerly in Matrix A in Matrix C.

This program includes a dummy option in the main menu which is labeled 5XXX. I have been considering adding an I/O routine so that the user didn't have to move back and forth between Stefan's program and this program, but I haven't done it yet. Including the dummy option means that if I decide to do it later I will simply have to replace the XXX with an appropriate nomenclature and proceed to enter the new option beginning at step N427 of the existing program. The result will be that I will not have to modify existing GTO addresses in the documentation of the present program when and if I decide to make an addition.

The functions of this program assume that the order of the matrix is in N. Any other control variables are calculated inside each individual function. This results in a substantial amount of duplicate code. Furthermore, since I was also trying to expand my programming skill on the HP35s I used different methods of calculating the same control variables.

Program Listing

This program uses a subroutine starting at M389 from Stefan's original Matrix Multi-Tool program. If you are using the revision dated April 3, 2008 then you must change the commands at locations N244 and N289 from XEQ M389 to XEQ M403.

```
N001   LBL N
N002   SF 10
N003   EQN 1NRM 2MOV 3MTH 4OPS 5XXX
N004   CF 10
N005   1
N006   X=Y?
N007   GTO N025
N008   X<>Y
N009   2
N010   X=Y?
N011   GTO N119
```

```
N012 x<>y
N013 3
N014 x=y?
N015 GT0 N222
N016 x<>y
N017 4
N018 x=y?
N019 GT0 N324
N020 x<>y
N021 5
N022 x=y?
N023 GT0 N427
N024 GT0 N002
N025 SF 10
N026 EQN 1ABS 2COL 3ROW
N027 CF 10
N028 1
N029 x=y?
N030 GT0 N040
N031 x<>y
N032 2
N033 x=y?
N034 GT0 N057
N035 x<>y
N036 3
N037 x=y?
N038 GT0 N084
N039 GT0 N002
N040 RCL N
N041 x^2
N042 1
N043 -
N044 1000
N045 /
N046 STO I
N047 0
N048 RCL (I)
N049 x^2
N050 +
N051 ISG I
N052 GT0 N048
N053 SQRTx
N054 STO A
N055 VIEW A
N056 GT0 N002
N057 RCL N
N058 1
N059 -
N060 1000
N061 /
N062 STO I
N063 STO J
N064 CLx
N065 ENTER
N066 0
N067 RCL (I)
N068 ABS
N069 +
N070 ISG I
N071 GT0 N067
N072 x<y?
N073 x<>y
N074 RCL N
N075 1000
N076 /
N077 STO+ I
N078 x<>y
N079 ISG J
N080 GT0 N066
N081 STO C
N082 VIEW C
N083 GT0 N002
N084 RCL N
N085 STO I
N086 1
N087 -
N088 1000
N089 /
N090 STOx I
N091 STO J
N092 RCL N
```



```
N093 100000
N094 /
N095 STO+ I
N096 CLX
N097 ENTER
N098 0
N099 RCL (I)
N100 ABS
N101 +
N102 ISG I
N103 GT0 N099
N104 x<y?
N105 x<>y
N106 RCL N
N107 x^2
N108 1
N109 -
N110 STO- I
N111 0.001
N112 STO+ I
N113 REGZ
N114 ISG J
N115 GT0 N098
N116 STO R
N117 VIEW R
N118 GT0 N002
N119 SF 10
N120 EQN 1STOC 4STOD 2RCLC 5RCLD 3EXCC 6EXCD 2 spaces between each
N121 CF 10
N122 1
N123 x=y?
N124 GT0 N146
N125 x<>y
N126 2
N127 x=y?
N128 GT0 N163
N129 x<>y
N130 3
N131 x=y?
N132 GT0 N180
N133 x<>y
N134 4
N135 x=y?
N136 GT0 N198
N137 x<>y
N138 5
N139 x=y?
N140 GT0 N206
N141 x<>y
N142 6
N143 x=y?
N144 GT0 N214
N145 GT0 N002
N146 0
N147 STO J
N148 RCL N
N149 x^2
N150 STO+ J
N151 1
N152 -
N153 1000
N154 /
N155 STO I
N156 RCL (I)
N157 STO (J)
N158 1
N159 STO+ J
N160 ISG I
N161 GT0 N156
N162 GT0 N002
N163 0
N164 STO J
N165 RCL N
N166 x^2
N167 STO+ J
N168 1
N169 -
N170 1000
N171 /
N172 STO I
N173 RCL (J)
```

```
N174 ST0 (I)
N175 1
N176 ST0+ J
N177 ISG I
N178 GT0 N173
N179 GT0 N002
N180 0
N181 ST0 J
N182 RCL N
N183 x^2
N184 ST0+ J
N185 1
N186 -
N187 1000
N188 /
N189 ST0 I
N190 RCL (I)
N191 x<>(J)
N192 ST0 (I)
N193 1
N194 ST0+ J
N195 ISG I
N196 GT0 N190
N197 GT0 N002
N198 RCL N
N199 ENTER
N200 x^2
N201 +
N202 1
N203 +
N204 ST0 J
N205 GT0 N148
N206 RCL N
N207 ENTER
N208 x^2
N209 +
N210 1
N211 +
N212 ST0 J
N213 GT0 N165
N214 RCL N
N215 ENTER
N216 x^2
N217 +
N218 1
N219 +
N220 ST0 J
N221 GT0 N182
N222 SF 10
N223 EQN 1AxB 2AxC 3A+C 4A - C
N224 CF10
N225 1
N226 x=y
N227 GT0 N241
N228 x<>y
N229 2
N230 x=y?
N231 GT0 N252
N232 x<>y
N233 3
N234 x=y?
N235 GT0 N301
N236 x<>y
N237 4
N238 x=y?
N239 GT0 N303
N240 GT0 N002
N241 RCL N
N242 ENTER
N243 x^2
N244 2
N245 x
N246 +
N247 ST0 E
N248 XEQ M389
N249 CF 0
N250 CF 2
N251 GT0 N002
N252 RCL N
N253 ENTER
N254 x^2
```

```
N255 x
N256 4
N257 /
N258 2
N259 +
N260 INTG
N261 STO T
N262 VIEW T
N263 RCL N
N264 ENTER
N265 x^2
N266 2
N267 x
N268 +
N269 STO E
N270 RCL N
N271 1
N272 -
N273 1000
N274 /
N275 STO V
N276 RCL N
N277 STO J
N278 x^2
N279 STO+ J
N280 STO+ J
N281 STO I
N282 STO+ I
N283 1
N284 STO- J
N285 STO- I
N286 -
N287 1000
N288 /
N289 STO+ I
N290 RCL (I)
N291 STO (J)
N292 DSE J
N293 DSE I
N294 GTO N290
N295 XEQ M389
N296 ISG V
N297 GTO N276
N298 CF 0
N299 CF 2
N300 GTO N180
N301 1
N302 GTO N304
N303 -1
N304 STO S
N305 RCL N
N306 x^2
N307 STO J
N308 1
N309 -
N310 1000
N311 /
N312 STO I
N313 RCL (I)
N314 RCL (J)
N315 RCL S
N316 x
N317 +
N318 STO (J)
N319 1
N320 STO+ J
N321 ISG I
N322 GTO N313
N323 GTO N180
N324 SF 10
N325 EQN 1DOT 2IDN 3FIL 4TRN
N326 CF 10
N327 1
N328 x=y?
N329 GTO N343
N330 x<>y
N331 2
N332 x=y?
N333 GTO N366
N334 x<>y
N335 3
```

```
N336 x=y?  
N337 GT0 N389  
N338 x<>y  
N339 4  
N340 x=y?  
N341 GT0 N401  
N342 GT0 N002  
N343 RCL N  
N344 x^2  
N345 STO J  
N346 1  
N347 -  
N348 1000  
N349 /  
N350 STO I  
N351 0  
N352 RCL (I)  
N353 RCLx (J)  
N354 +  
N355 1  
N356 STO+ J  
N357 R Down  
N358 ISG I  
N359 GT0 N352  
N360 ENTER  
N361 SF 10  
N362 EQN DOT =  
N363 CF 10  
N364 STOP  
N365 GT0 N002  
N366 RCL N  
N367 x^2  
N368 1  
N369 -  
N370 1000  
N371 /  
N372 STO I  
N373 RCL I  
N374 INTG  
N375 RCL N  
N376 1  
N377 +  
N378 /  
N379 FP  
N380 x=0?  
N381 GT0 N384  
N382 0  
N383 GT0 N385  
N384 1  
N385 STO (I)  
N386 ISG I  
N387 GT0 N373  
N388 GT0 N002  
N389 INPUT K  
N390 RCL N  
N391 x^2  
N392 STO I  
N393 1  
N394 STO- I  
N395 RCL K  
N396 STO (I)  
N397 RCL I  
N398 x not equal 0 ?  
N399 GT0 N393  
N400 GT0 N002  
N401 RCL N  
N402 x^2  
N403 STO W  
N404 STO J  
N405 1  
N406 -  
N407 1000  
N408 /  
N409 STO I  
N410 RCL (I)  
N411 STO (J)  
N412 RCL N  
N413 STO+ J  
N414 x^2  
N415 2  
N416 x
```

```
N417 RCL J
N418 x<y?
N419 GT0 N424
N420 1
N421 STO+ W
N422 RCL W
N423 STO J
N424 ISG I
N425 GT0 N410
N426 GT0 N180
N427 GT0 N002
```

Length = 1532; Checksum = 0AE0

Since some users have reported the length and checksum may not be reliable I have provided some example problems which will demonstrate that the program has been properly entered. We will use three 3x3 matrices

HP33			HP15			Product		
23	15	17	33	16	72	263	150	512
8	11	-6	-24	-10	-57	48	42	51
4	15	12	-8	-4	-17	-324	-134	-771

The matrix at the left is from the demonstration of the Solutions of Simultaneous Equations program on page 1-18 of the *hp33s user's guide*. The matrix in the center is from the demonstration of a residual correction program on page 121 of the *HP-15C Advanced Functions Handbook*. The matrix on the right is the product of the matrix on the left and the matrix in the center; i.e., HP33xHP15

1. Use Stefan's program to enter HP15 into Matrix A..
2. Use the NRM functions of this program to calculate the Frobenius norm (ABS) of HP15 as 104.033648403, the row norm (ROW) as 121 and the column norm (COL) as 146 .
3. Use the STOC and STOD functions of this program to store HP15 into Matrix C and into Matrix D. This leaves HP15 in all three matrix locations.
4. Use Stefan's program to enter HP33 into Matrix A.
5. Use the DOT function of this program to calculate the dot product of HP33 and HP15 as 1967.
6. Use the AxC function of this program to calculate the product HP33xHP15. The execution time estimate will be 8 seconds. The product will be in Matrix A and HP33 will be in Matrix C.
7. Use the ABS function of this program to calculate the Frobenius norm of HP33xHP15 as 1,038.197958 .If this is correct you can be reasonably confident that the AxC function is working properly. You can verify the individual elements of the product by returning to Stefan's program.
8. Use the EXCC function of this program to exchange the contents of Matrix A and the contents of Matrix C. This places HP33 in Matrix A and the product in Matrix C.
9. Use the STOD function of this program to store the contents of Matrix A into Matrix D. This places HP33 in Marix D.
10. Use Stefan's program to enter the vector [31 17 14] in b. Solve and read the determinant of HP33 as 4,598. Read the solution vector X to the problem on page 15-18 of the *hp33s user's guide* from b as

9.30622009565E-1
7.94258373205E-1

-0.13636363636

The inverse of HP33 will be in matrix A and in Matrix C. HP33 will still be in Matrix D.

11. Use the RCLD function of this program to recall HP33 from Matrix D and place it into Matrix A. Use the AxB function of this program to multiply HP33 by the solution vector found in step 10. The product HP33xX will be in b and in the left hand column of Matrix C.

12. Use Stefan's program to read out the product HP33xX as

31
17
14.0000000001

13. Use the STO C function of this program to store the contents of Matrix A (HP33) into Matrix C. Use the IDN function of this program to place the 3x3 identity matrix in Matrix A. Use the A+C function of this program to place the sum HP33 and the identity matrix in Matrix A. The identity matrix will still be in matrix C. Use the ABS function of this program to calculate the Frobenius norm of the sum of HP33 and the identity matrix as 41.7612260356. Again, if this is correct you can be reasonably confident that the IDN and A+C functions are working properly. You can verify the individual elements of the sum with Stefan's program to be

24	15	17
8	12	-6
4	15	13

14. Use the STOC function of this program to store the matrix in Matrix A into Matrix C. Use the FIL function of this program to place a matrix of all two's in Matrix A. Use the EXCC function of this program to exchange the matrices in Matrix A and Matrix C. Use the A-C function of this program to subtract the matrix of all two's from the matrix in Matrix A (which is the sum of HP33 and the 3x3 identity matrix) in Matrix A.. Calculate the Frobenius norm of the result as 37.0405183549. Once again, if this is correct you can be reasonably confident that the FIL and A-C functions are working properly. You can verify the individual elements of the difference with Stefan's program to be

22	13	15
6	10	-8
2	13	11

15. Use the EXCD function of this program to bring HP33 to Matrix A. Use the TRN function of this program to place the transpose of HP-33 in Matrix A. Use Stefan's program to verify that the transpose of HP33 is in Matrix A.

Some other subroutine capabilities of Stefan's program:

Entry at M328 will copy the contents of the temporary matrix into matrix A and stop with the contents of the D register in X.

Entry at M383 will stop with a time estimate in the display. When R/S is pressed the product of matrix A and vector B will be placed in the left hand column of the temporary matrix and in vector b. To avoid stopping for the time estimate enter at M389.

Entry at M415 will copy the contents of the left hand column of the temporary matrix into vector b.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

A "Cadillac " Quadratic Solver for the hp 33s and HP35s

Posted by [Palmer O. Hanson, Jr.](#) on 29 Apr 2008, 11:40 a.m.

NOTE: The program for the hp33s was previously published as Article 593 on 10 March 2006. The program sequence had errors at A0008, B0022 and E0009. I was unable to correct those errors because I did not have a valid password. This article repeats the hp 33s material with the errors corrected and adds a similar program for the HP35s.

The Polynomial Root Finder program in pages 15-20 through 15-32 of the *hp 33s user's guide* offers a quadratic solution as an option. That routine is a "garden variety" solution which is adequate for most quadratic problems; however, users may occasionally encounter quadratic problems which cannot be correctly solved with that routine. An example is a problem for which the discriminant is erroneously found to be zero because the two parts of the discriminant have more significant digits than the number of significant digits carried by the machine on which the quadratic solver is installed.

A quadratic solver which appears on pages 208-211 of the *HP-15C Advanced Functions Handbook* uses a so-called "tricky property of the SUM + and SUM - commands" in a way that is equivalent to carrying twenty significant digits in the calculation of the discriminant. As I was working my way through the *hp 33s user's guide* I found the following statement on page 1-19: "... During some complicated internal calculations, the calculator uses 15-digit precision for immediate results. ..." That sounded a lot like the "tricky properties". I did some tests that convinced me that the "tricky properties" were available in the hp 33s. I then decided to try to translate the HP-15C program for use with the hp 33s. Steps 011 through 025 of the HP-15C program are

```
011 CLEAR SUM
012 RCL 8
013 STO 7
014 RCL / I
015 RND
016 RCL I
017 SUM -
018 RCL 9
019 x <> 7
020 x <> y
021 RCL 8
022 SUM -
023 R Down
024 SUM -
025 RCL 7
```

and so on where there is an additional SUM - at step 052 and there are no SUM + commands in the loop. That means that the HP-15C is able to accept inputs to the statistics registers which would cause the number of entries (n) to become negative. The statistics routines in the HP-41, the HP-32S and several other HP machines can also do that. The statistics routine in the hp 33s cannot. After a little thought it occurred to me that hp 33s routine may not care how the value for the number of entries (n) had been accumulated in register 28. To prove that I did a CL SUM and followed it

by three SUM + commands where 2 was in the y register and 3 was in the x register. I recalled the means and saw the values 3 and 2 as expected. Then I stored 28 (the number of the statistics register which holds the value of n) in I, entered a 3 in the display, and did a STO+ (i). I recalled the means and saw the values 1.5 and 1.0 . To make a long story short I was able to successfully translate the HP-15C program for use on the hp 33s by storing a four for n in statistics register 28 after the CL SUM command and before the SUM - commands. The program which follows implements the HP-15C program on the hp 33s. Actually, it is not a direct translation from the HP-15C program but rather is a translation from my translation of the HP-15C program for use on the HP-41 as described in my submission "Update: 13 Digit Precision on the HP-15C" of 23 July 2003 which appears in Archive 13 of the Forum.

The HP-33s and HP-35s programs below accept the coefficients A, B, and C of the quadratic $Ax^2 + Bx + C = 0$. For complex roots Flag zero is set and the imaginary part is in the upper part of the display. The entered value for B is divided by -2 to convert it for use by the solution routine which is for the modified quadratic $Ax^2 - 2Bx + C = 0$. This is similar to the quadratic solution method on pages 110-111 of the *HP-27 Owner's Handbook*. The value of A does not have to be a value of one as with the quadratic routine from the *hp 33s Users Guide*. Some test cases follow the program listing . You will find that the quadratic program from the *hp 33s user's guide* fails to get the correct answer for several of the cases. The translated program gets the correct answer in all eight cases.

HP-33S Program Listing

```

H0001 LBL H
H0002 INPUT A
H0003 STO D
H0004 INPUT B
H0005 -2
H0006 /
H0007 STO F
H0008 STO H
H0009 INPUT C
H0010 STO G
H0011 STO I
H0012 CF 0
H0013 SCI 2
A0001 LBL A
A0002 CL SUM Clear statistics registers
A0003 28
A0004 STO i
A0005 4
A0006 STO(i)
A0007 33
A0008 STO i
A0009 RCL H
A0010 STO(i)
A0011 RCL D
A0012 /
A0013 RND
A0014 RCL D
A0015 SUM -
A0016 RCL I
A0017 SUM xy Recall statistics product from register 33
A0018 x<>y
A0019 STO(i)
A0020 R DOWN Roll down
A0021 x<>y
A0022 RCL H

```

A0023 SUM -
A0024 R DOWN
A0025 SUM -
A0026 SUM xy
A0027 ABS
A0028 RCL I
A0029 ABS
A0030 $x \leq y$? Is x less than or equal to y
A0031 GTO B
A0032 ENTER
A0033 R UP Roll up
A0034 STO H
A0035 SUM xy
A0036 STO I
A0037 ABS
A0038 1 E 20
A0039 X
A0040 RCL G
A0041 ABS
A0042 $x \leq y$?
A0043 GTO A
B0001 LBL B
B0002 FIX 9
B0003 RCL H
B0004 x^2
B0005 STO(i)
B0006 RCL D
B0007 RCL I
B0008 SUM -
B0009 RCL F
B0010 SUM xy
B0011 $x < 0$?
B0012 GTO E
B0013 SQR x Square root of x
B0014 RCL F
B0015 SGN
B0016 X
B0017 +
B0018 STO J
B0019 $x < > y$
B0020 /
B0021 RCL J
B0022 $x < > 0$? x is not equal to zero
B0023 GTO F
B0024 CLx
B0025 STOP
F0001 LBL F
F0002 RCL G
F0003 $x < > y$
F0004 /
F0005 STOP
E0001 LBL E
E0002 SF 0
E0003 +/-

E0004 SQR x
E0005 RCL D
E0006 /
E0007 $x \leftrightarrow y$
E0008 R UP
E0009 /
E0010 STOP

HP35S Program Listing

The hp 33s program listing above requires a number of changes for use on the HP35s; for example, to account for the different assignment of memory locations for the statistics registers A0003 must be changed from +28 to -27 and A0007 must be changed from +33 to -32. Changes are also required to account for the difference between i and (i) with the hp 33s and I and (I) with the HP35s. It would also be necessary to assign a different variable name wherever I is used in the HP 33s program. Finally, the hp 33s version uses five labels because direct address transfers are not available. The result of all of this is that I rewrote the hp 33s program for the HP35s using direct addressing to save labels, and in the process made the other necessary changes.

Q001 LBL Q
Q002 SF 10
Q003 EQN $Ax^2 + Bx + C$
Q004 CF 10
Q005 INPUT A
Q006 STO D
Q007 INPUT B
Q008 -2
Q009 /
Q010 STO F
Q011 STO H
Q012 INPUT C
Q013 STO G
Q014 STO K
Q015 CF 0
Q016 SCI 2
Q017 CL SUM Clear statistics registers
Q018 -27
Q019 STO I
Q020 4
Q021 STO (I)
Q022 -32
Q023 STO I
Q024 RCL H
Q025 STO (I)
Q026 RCL D
Q027 /
Q028 RND
Q029 RCL D
Q030 SUM -
Q031 RCL K
Q032 SUM xy Recall statistics product from register 33
Q033 $x \leftrightarrow y$
Q034 STO (I)
Q035 R DOWN Roll down
Q036 $x \leftrightarrow y$

Q037 RCL H
Q038 SUM -
Q039 R DOWN
Q040 SUM -
Q041 SUM xy
Q042 ABS
Q043 RCL K
Q044 ABS
Q045 $x \leq y$? Is x less than or equal to y
Q046 GTO Q059
Q047 ENTER
Q048 R UP Roll up
Q049 STO H
Q050 SUM xy
Q051 STO K
Q052 ABS
Q053 1 E 20
Q054 X
Q055 RCL G
Q056 ABS
Q057 $x \leq y$?
Q058 GTO Q017
Q059 FIX 9
Q060 RCL H
Q061 x^2
Q062 STO (I)
Q063 RCL D
Q064 RCL K
Q065 SUM -
Q066 RCL F
Q067 SUM xy
Q068 $x < 0$?
Q069 GTO Q087
Q070 SQR x Square root of x
Q071 RCL F
Q072 SGN
Q073 X
Q074 +
Q075 STO L
Q076 $x < > y$
Q077 /
Q078 RCL L
Q079 $x < > 0$? x is not equal to zero
Q080 GTO Q083
Q081 CLx
Q082 STOP
Q083 RCL G
Q084 $x < > y$
Q085 /
Q086 STOP
Q087 SF 0
Q088 +/-
Q089 SQR x
Q090 RCL D

Q091 /
Q092 $x < > y$
Q093 R UP
Q094 /
Q095 STOP

TEST CASES

The *HP-15C Advanced Functions Handbook* describes the program as "like Grandmother's expensive chinaware, reserved for special occasions." I suspect that may be because it uses more memory and much more time to run. For the test cases below the hp 33s never uses more than a second or two to reach a solution. The HP35s runs somewhat longer.

Cases 1 through 4 are equivalent to cases 3 through 6 on page 207 of the *HP-15C Advanced Functions Handbook* . Cases 5 and 6 and are equivalent to the problems defined on page 211 of *HP-15C Advanced Functions Handbook*. Cases 7 and 8 are two additional problems that I devised which require a 15 digit capability if the discriminant is to be calculated correctly.

Case 1:

$$a = 1\text{E-}13 \quad b = -2 \quad c = 1$$

$$R1 = 2\text{E}13 \quad R2 = 0.5$$

Case 2:

$$a = 654,323 \quad b = -1,308,644 \quad c = 654,321$$

$$R1 = 1 \quad R2 = 0.999996943\dots$$

Case 3:

$$a = 11,713 \quad b = -1,470,492 \quad c = 46,152,709$$

$$\text{Re} = 62.77179203\dots ; \text{Im} = 8.5375\text{E-}05$$

Case 4:

$$a = 80,841 \quad b = -1,975,288 \quad c = 12,066,163$$

$$\text{Re} = 12.21711755\dots ; \text{Im} = 0.001374514\dots$$

Case 5:

$$a = 4,877,361,379 \quad b = -9,754,525,226 \quad c = 4,877,163,849$$

$$\text{Re} = 0.999979750 ; \text{Im} = 2.8995463\text{E-}10$$

Case 6:

$$a = 1 \quad b = -222,223 \quad c = 12,193,329,370$$

$$R1 = 123,458 ; R2 = 98765$$

Case 7:

$a = 11,111,119$ $b = -22,222,222$ $c = 11,111,103$

$R1 = 1$; $R2 = 0.999998560001$

Case 8::

$a = 8,441,600$ $b = -22,222,222$ $c = 14,624,809$

$Re = 1.31623282316$ $Im = \pm 1.05290400129E-6$

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

The MoHPC 9114 Repair Manual

Posted by [PeterP](#) on 26 Apr 2008, 8:37 p.m.

[Click for full document](#)

<http://www.hpmuseum.org/guest/peterp/9114repm.pdf>

The article at the above link is a write-up of the collective wisdom that I found in the archives when I was repairing my 9114 (under the wonderful tutelage of Tony Duell). It also includes some links to important information on the net with regards to the 9114 disk drives, like datasheets, where to buy replacement batteries, etc.

It is an attempt to combine the large amount of knowledge with regards to the repair of those disk-drives in one convenient location.

The order of information presented tries to follow the frequency of particular questions asked as well as the usual step by step test procedure the people knowledgeable about the 9114 suggest.

The appendix was supposed to contain the full document from Tony Duell with pictures on how to clean the 9114. Unfortunately Tony's article is too big (4mb) so it is not included for now. I am looking for ways to include it and when I find a way I will amend this document)

While there are numerous consistent providers of advice on the 9114 and I have tried to mention them at the end of this document, I ask for forgiveness if I have overlooked to give credit to certain people. It is solely due to my own limitations.

Also, while I provided some close-up pictures, no real knowledge provided in this document is from myself. Yet as I am a beneficiary of the information provided by others I want to find some way to give back something useful as well.

Index

- How to find all threads with regards to the 9114 on the MoHPC website
- Commands to test/use the 9114
- The battery pack
- Replace the battery pack
- Open the battery pack
- Make a direct power supply for the 9114
- The micro-fuse of the battery pack
- Schematics for the charger
- What floppy disks to use with the 9114
- How to open the 9114
- Repairing the eject mechanism
- Replacing the drive
- Repairing/realigning the drive head
- Pictures of head & drive
- Testing of the controller board
- Schematics of the controller board

- Main contributors
- Tony Duell's article on cleaning the eject mechanism of the 9114

Click for full document
<http://www.hpmuseum.org/guest/peterp/9114repm.pdf>

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP Calculator Simulators - with downloads

Posted by [Mike T.](#) on 5 Apr 2008, 5:08 p.m.

This article contains links to the latest versions of my HP Calculator Simulators including the HP25C, HP29C, HP31E, HP33C, HP67A and now HP32E.



Latest Status - 04 Aug 13 - LINUX!

I've managed to work out how to draw a decent representation of the keyboard and display using just X11, which wasn't as difficult as I thought once I'd decided to just assume that output device supported 24-bit colour. I'm now trying to figure out the simplest way to code up the rest of the functionality I want, and will probably use double precision floating point numbers initially and switch to using the decnumber library later. My original Visual BASIC code was very heavily dependent on using and manipulating variable length strings, so the techniques I used do not translate to C easily and when it comes to C I'm still learning what 'works' and what doesn't...

Latest Status - 30 Mar 13

I haven't done much with any of these emulators for a while, as I'm trying to work out how to draw the custom buttons, and other controls using GTK2 so I can port them to Linux as I don't use windows on my own machines very often any more. However I would still like to finish off the HP34C emulator if I could work out how to get the answers for intergrate and solve that were very close to the real machine.

Latest Status - 01 Sep 11

Successfully checked that the simulators work on Windows 7, once the necessary runtime support is installed (MSVBVM50.EXE).

Status Update - 17 Mar 10

Unfortunately I've been unable to spend much time on this project recently and what little time I have spent has been on developing the prototypes described below and trying to find and elegant ways of structuring the code that 'works'.

Status Update - 22 Oct 09

Need to test my code to preform intergration using the Romberg Method - but I'm a bit stuck for ideas on how to approach the 'solve' function that would allow me to finish the HP34C simulation.

Status Update - 03 May 09

I have now successfully implemented the statistical routines I needed to approximate the normal and inverse normal distribution functions so I am finally able to release a simulation for the HP32E along with some minor fixes to the other simulators. I'm particularly pleased to be able to add this simulation to my collection, as the HP32E was one of the first HP calculators I owned and it's ability to evaluate the normal and inverse normal distribution functions accurately when everyone else was having to use look up tables was quite something at the time.

Unfortunately I've not been able to address any of the known bugs in the HP67A simulation in this release. Sorry, but I just have not had enough time - I have not forgotten them and will fix them at some point.

I'm still restructuring code to make it more object orientated, however this latest release is based on the older code base as I've not been able to spend much time on this little project recently...

Simulations

Finished simulators for the HP21A, HP25C, HP29C, HP31E, HP32E, HP33C, and HP67A are included below along with screen shots for other models that are under development including the HP10C, HP11C, HP22A, HP27A and HP34C.

- HP10C [screenshot](#)
- HP11C [screenshot](#)
- HP21A [screenshot](#) [program source code](#) (updated - 03 May 09)
- HP22A [screenshot](#)
- HP25C [screenshot](#) [program source code](#) (updated - 03 May 09)
- HP27A [screenshot](#)
- HP29C [screenshot](#) [program source code](#) (updated - 03 May 09)

- HP31E [screenshot program source code](#) (updated - 03 May 09)
- HP32E [screenshot program source code](#) (updated - 03 May 09)
- HP33C [screenshot program source code](#) (updated - 03 May 09)
- HP34C [screenshot](#)
- HP67A [screenshot program source code](#) (updated - 03 May 09)

Prototypes

Having read some items in the forum on this web site regarding reprogramming the HP20B, I decided to test my new code using a hypothetical simulation based on the HP20B form factor and display size. This has allowed me to work out what seems to work and what doesn't, so it is now time to go back to the drawing board again and reassess what sort of object model I want given the constraints imposed by the language and environment. Work on the 'new' simulation below is still at a very early stage and is far from complete though I have implemented most of the basic functions to allow you to play around with it a bit.

- Hypothetical HP32B [screenshot source code](#) (updated - 18 Nov 08)

I really like the two line display and I'm reasonably happy with the layout of the numeric keypad and the main functions - but I'd be interested to see what other suggestions anyone else has that you think would work better...

- Hypothetical HP34B [screenshot source code](#) (new - 18 Nov 08)

I noticed that it might be possible to accommodate all the HP34C functions using just two function keys on the HP20B keyboard, however it is necessary to use all possible shifted functions to allow for a PRGM/RUN and ON/OFF switches. It may not of course be possible to do this on a real HP20B by modifying just the firmware - in which case something will have to give, but this should give you some idea what could be achieved...

When I've finished restructuring the code I'll then concentrate on rewriting the existing simulations to use the new object orientated code. Hopefully I will be able to fix the outstanding bugs in the HP10C and HP67 simulators and finish off the simulators for the HP34C, and HP11C.

Known Issues

- There are a couple of minor problems with the HP67 simulation that I haven't fixed yet, however the HP21A, HP29C, HP31E, HP32E and HP33C simulations should behave as expected. (Though the HP32E simulation does allow the use of R9 which was not available on the real thing).
- Since the simulators do not use decimal arithmetic, the numeric accuracy depends on the accuracy of the floating point arithmetic routines used and may differ very slightly from the real thing.

It is probably worth checking this page for updates regularly as I will post new builds here whenever I fix any reported bugs.

Instructions

To run any of the simulators extract the contents of the ZIP file to a folder and just double click on the appropriate executable. They will run on Windows NT 4.0¹ (but NOT Windows 95/98/Me) and the 32-bit versions of Windows

2000, Windows XP and Windows Vista, providing that the Visual BASIC 5.0 runtime is installed.

Help Wanted

- If you notice that any of these simulators don't behave in the same way as the real thing let me know, your feedback is very useful and helps improve the quality of the simulators. Even if it isn't a bug let me know what you think.
- I need to work out how to implement integrate and solve in order to be able to complete my collection of SPICE machines with an HP34C, but I need some help in order to understand how these should be implemented in order to be able to get the same result as the real thing.
- Visual BASIC does have some financial functions, but I don't understand them well enough to understand how I could use them to implement the financial functions on the HP12C, HP22A, and HP27A.

Mike T.

¹ Windows NT 4.0 can only run a maximum of three different simulators at any time.

Edited: 4 Aug 2013, 7:05 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Effective Computer-aided Calculator Programming - Part 1 - Voyager

Posted by **Egan Ford** on 2 Apr 2008, 12:47 a.m.

Abstract

Keystroke programming for small *quick&dirty* programs it is very efficient, but what about for long or very long programs? Debugging a 100+ instruction program can be a challenge, especially when only one instruction is visible. Many of us can write RPN code in text editors, and then run the program in our heads before keying the sequence. While this may work for some, there are more effective ways.

The purpose for this series of articles is to describe the options available to aid in the rapid development and testing of RPN code by leveraging the use of text editors, compilers, and accelerated simulators and emulators.

Part 1 of this series supports the follow models: 11C, 12C, 15C, and the 16C.

Audience

Windows, Linux, and OS/X users that want to save time debugging and testing RPN programs; especially long complex competition and challenge code.

Prerequisites

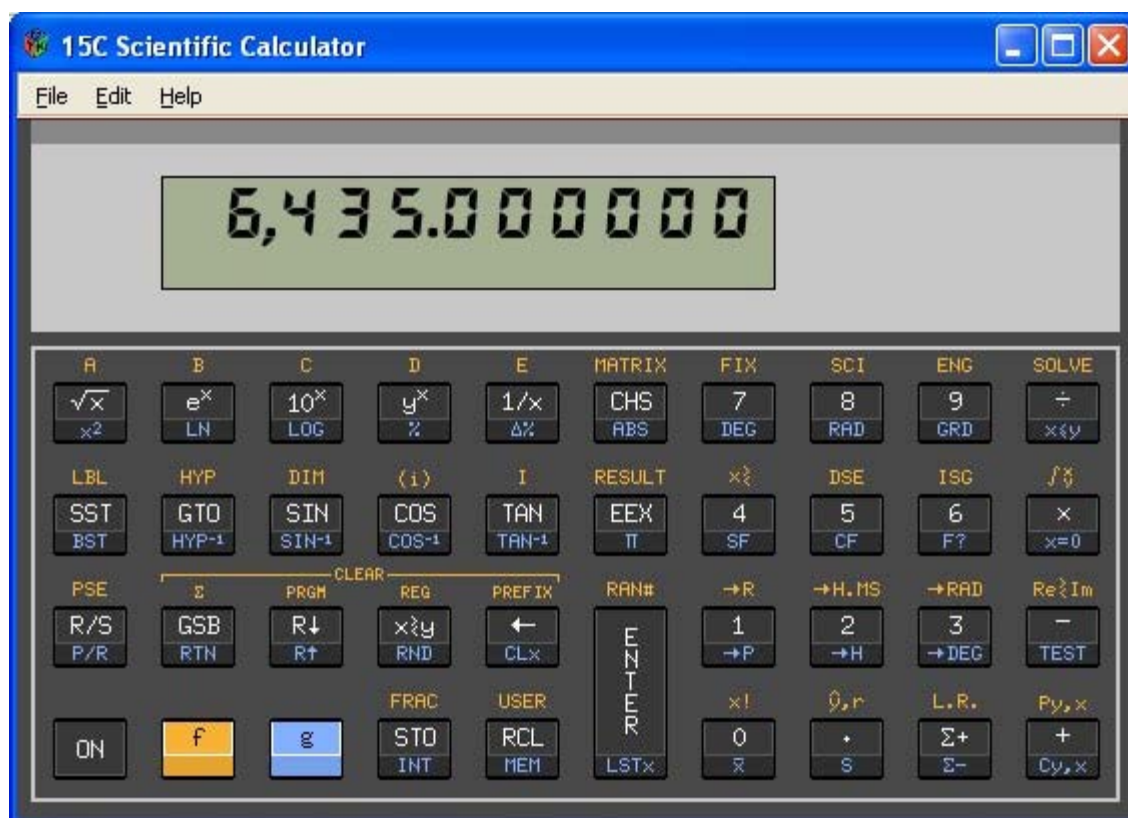
Perl

The Voyager compiler used in this article requires a Perl interpreter. For Linux and OS/X nothing is required, however for Windows, Perl must be installed. There are a number of freely available Perl implementations for Windows; one of the most popular is the freely available ActivePerl (<http://www.activestate.com/store/activeperl/download/>).

To verify that Perl is working open up a command prompt and type:

```
perl -e "print \"hello world\n\""
```

Nonpareil



Nonpareil 15C Windows Snap Shot

Nonpareil is multiplatform (Windows, Linux, OS/X, and UNIX) multicalculator emulator that supports many classic models including all the Voyagers (sans the 10C). Since the emulator uses the original Voyager ROMs the numeric results will be exactly the same.

NOTE: The Voyager compiler in this article only supports Nonpareil.

To install Nonpareil for Windows visit <http://nonpareil.brouhaha.com/download> and download and install GTK first, then download and extract the Nonpareil Windows zip file into any directory. To launch Nonpareil, change to the nonpareil-0.77-win32 directory and type:

```
nonpareil calculator
```

e.g.

```
nonpareil 15c
```

Alternatively you can give no argument and then be greeted by a GUI calculator selector.

To install Nonpareil for Linux/Unix first verify that GTK is installed, and then download the source from <http://nonpareil.brouhaha.com/download>, extract, and type as root:

```
cd nonpareil-0.78/scons-local
tar -xzf scons-local-0.96.90.tar.gz
cd ..
scons-local/scons.py install
```

This will place nonpareil in /usr/local/bin. If you wish to install in a different directory read the INSTALL file for more details.

OS/X users can get Nonpareil from <http://homepage.mac.com/mba/nonpareil/>. This version unfortunately does not save or read Nonpareil state files making the compiler only good for syntax checking, macros, virtual labels (12C), numeric expansion, and publication of code (pretty print). OS/X users may opt for VMware Fusion, Parallels, or Wine with the

Windows or Linux version of Nonpareil.

Voyager Compiler

vcomp.pl is a Nonpareil Voyager program compiler and lister. vcomp.pl achieves this by reading and writing Nonpareil state files, i.e. the files that contain the RAM contents of each emulated calculator.

vcomp.pl has a number of advanced features to make Voyager development and testing more efficient:

1. Syntax checking with support for user definable statements.
2. Numeric expansion. Multiple digit constant numbers can use a single line of code.
3. Duplicate label checking. Although legal, warnings are issued when duplicate labels are encountered.
4. GTO checking. No point in going nowhere.
5. Macro expansion. Macros can be defined on a per calculator basis.
6. Virtual 12C labels. LBL can be use with 12C programs. All LBL and GTO statements will be properly converted to line numbers when compiled.
7. Pretty print listings than can contain line numbers, statements, and/or keystroke sequences. Listings can be displayed using multiple columns to minimize space usage when publishing your code.
8. An internal dictionary of statements and keystrokes that can be listed and used as a reference guide.

Windows Install

The Windows version can be obtained from here: <http://www.hpmmuseum.org/guest/eganford/vcomp.zip>, to install just copy into any directory in your path.

Linux, OS/X, Cygwin, Unix, etc... Install

The non-Windows version can be obtain from here: <http://www.hpmmuseum.org/guest/eganford/vcomp.gz>, to install just copy into any directory in your path and type:

```
gunzip <vcomp.gz >vcomp.pl
chmod 755 vcomp.pl
```

Using vcomp.pl

vcomp.pl without any options will produce the following helpful output:

```
vcomp.pl dict [-help] [-c|-cc n] <model>
          list [-help] [-c|-cc n] [-code|-linecode] <file>
          comp [-help] [-strip n] [-phase 1|2] [-c|-cc n] [-new <model>] -nst <file> <code>

<model>  Must be 11C, 12C, 15C, or 16C.
<file>   Is an existing or new Nonpareil State file.
<code>   Is your text source code.
```

vcomp.pl has three modes of operation: **dictionary**, **list**, and **compile**. Although all the examples use the first four letters, the lazy (or efficient) can opt to use just the first letter when specifying the mode of operation. Each mode has options specific to that mode. However there are few universal options.

The universal option *-help* can be used with any vcomp.pl command/mode to display usage and examples. E.g.:

```
vcomp.pl dict -help
```

will output:

```
vcomp.pl dict [-c|-cc n] <model>

-c n      Print output in n columns.
-cc n     Print output using x columns that can fit in n characters.

<model>   Must be 11C, 12C, 15C, or 16C.
```

E.g. 11C dictionary in 2 columns:

```
vcomp.pl dict -c 2 11C
```

E.g. 15C dictionary in x columns that can fix in 80 characters:

```
vcomp.pl dict -cc 80 15C
```

-c and -cc are two other universal options that can be used to columnize any output. -c n will display the output in n columns whereas -cc n will determine based on the length of the longest line how many columns can fit within an n character line, e.g. -cc 80 will columnize the output into m columns such that the maximum character count for any output line will <= 80.

Dictionary

The dictionary is used to convert statements to byte code (compiling) and byte code to statements (listing). The vomp.pl dict command can be used to list the contents of the dictionary, e.g.:

```
vcomp.pl dict -c 2 15C
```

Output:

%	43 14	RCL/ (i)	- 45,10,24
+	40	RCL/ .0	- 45,10,.0
-	30	RCL/ .1	- 45,10,.1
->DEG	43 3	RCL/ .2	- 45,10,.2
->H	43 2	RCL/ .3	- 45,10,.3
.			
.			
RCL- B	- 45,30,12	x^2	43 11
RCL- C	- 45,30,13	y^,r	42 48
RCL- D	- 45,30,14	y hat,r	
RCL- E	- 45,30,15	LIN EST	
RCL- I	- 45,30,25	y^x	14

The listing is sorted alphabetically. Nested statements (e.g. y hat,r, LIN EST above) are aliases. That is LIN EST could be used in your source instead of or in addition to y^,r when compiling, however when listing only the primary statement (e.g. y^,r) will be used. The dictionary and its contents are user definable.

Editing the Dictionary

To edit the dictionary for any model use your favorite text editor and search for "model Opcode Start", e.g. "11C Opcode Start", then scroll down until you find the statements you wish to edit or rearrange. E.g.:

ee	L.R.	42 49
ef	y^,r	42 48
ef	y hat, r	42 48
ef	LIN EST	42 48
f0	0	0

Notice how y^,r, y hat, r and LIN EST all have the same byte code. The first one listed will be the primary and the others will be aliases. Feel free to edit this to suit your taste, but make a backup first, just in case.

List

```
vcomp.pl list
```

will list the contents of any existing or newly created Nonpareil state file, e.g.:

```
vcomp.pl list -c 2 "c:\Documents and Settings\Egan\nonpareil\12C.nst"
```

Output:

01 -	0	0	24 -	1	1
02 -	44	2 STO 2	25 -	40	+
03 -	1	1	26 -	1	1
04 -	44	1 STO 1	27 -	4	4
05 -	44	0 STO 0	28 -	8	8
06 -	1	1	29 -	6	6
07 -	16	CHS	30 -	3	3
08 -	44 20	1 STOx 1	31 -	5	5
09 -	2	2	32 -	6	6
10 -	44 40	0 STO+ 0	33 -	4	4
11 -	45	0 RCL 0	34 -	8	8
12 -	1	1	35 -	20	x
13 -	1	1	36 -	5	5
14 -	21	y^x	37 -	0	0
15 -	22	1/x	38 -	5	5
16 -	45	1 RCL 1	39 -	48	.
17 -	20	x	40 -	2	2
18 -	44 40	2 STO+ 2	41 -	1	1
19 -	42 14	RND	42 -	10	/
20 -	43 35	x=0	43 -	1	1
21 -	43,33 23	GTO 23	44 -	1	1
22 -	43,33 06	GTO 06	45 -	22	1/x
23 -	45 2	RCL 2	46 -	21	y^x

The *-code* and *-linecode* options can be used to limit the amount of information, e.g.:

```
vcomp.pl list -c 5 -code 12C.nst
```

0	RCL 0	GTO 23	5	1
STO 2	1	GTO 06	6	/
1	1	RCL 2	4	1
STO 1	y^x	1	8	1
STO 0	1/x	+	x	1/x
1	RCL 1	1	5	y^x
CHS	x	4	0	
STOx 1	STO+ 2	8	5	
2	RND	6	.	
STO+ 0	x=0	3	2	

```
vcomp.pl list -c 3 -linecode 12C.nst
```

01 0	17 x	33 4
02 STO 2	18 STO+ 2	34 8
03 1	19 RND	35 x
04 STO 1	20 x=0	36 5
05 STO 0	21 GTO 23	37 0
06 1	22 GTO 06	38 5
07 CHS	23 RCL 2	39 .
08 STOx 1	24 1	40 2
09 2	25 +	41 1
10 STO+ 0	26 1	42 /
11 RCL 0	27 4	43 1
12 1	28 8	44 1
13 1	29 6	45 1/x
14 y^x	30 3	46 y^x
15 1/x	31 5	
16 RCL 1	32 6	

Compile

```
vcomp.pl comp
```

will compile plain text statements into Voyager byte code in the form of a Nonpareil state file.

When compiling you have the option of creating a new out-of-the-box state file or inserting your code into an existing state file preserving the contents of the stack and storage registers.

NOTE: Inserting into an existing state file is experimental. Backup your data first.

NOTE: New state files are not exactly "out-of-the-box", the 11C, 12C, and 15C are out-of-the-box + FIX 9.

Source Code Anatomy

- One statement per line, e.g. 12C program to calculate area of a circle:

```
;Area of a circle
ENTER
ENTER
*
3.141592654
*
```

- Comments start with semicolon (;). Any text right of any ";" will be ignored, e.g.:

```
;my blah program
LBL A          ;LBL A
2
*              ;X * 2 -> X
;Fix this bug later
```

- Numeric constants can be coded as a single statement, e.g.:

```
123
-123
-123.45
-123.45E-5
```

- Statements are case and white space insensitive, e.g.:

```
lbl a
LBL A
Lb1A
    LBL          A
```

are all the same statement.

- Macros start with &, e.g.:

```
&MOD
```

- Virtual labels (12C only) must start with a letter and can be any length, e.g.:

```
LBL A
LBL F00
LBL B2
GRO A
GTO F00
GTO B2
```

Compiler Example

To compile the source file area.12c and insert into an existing 12C.nst state file:

```
vcomp.pl comp -nst 12C.nst area.12c
```

To compile the source file area.12c and create a new 12C.nst state file:

```
vcomp.pl comp -new 12C -nst 12C.nst area.12c
```

The output for any successful compile should look like this:

```
Phase 1: Preprocessor
Phase 2: Check GTOS
Phase 3: Compile
Done!
```

The only difference between the two is the use of the *-new* option. This is required to instruct the compiler as to what model you are compiling for. Without *-new* the compiler can autodetect the model from the provided state file.

Because of the advanced preprocessing features of this compiler the post compiled listing may not match your source code, e.g.

Input file area.12:

```
ENTER
ENTER
*
3.141592654
*
```

Listing:

```
vcomp.pl list -linecode 12c.nst
```

```
01 ENTER
02 ENTER
03 x
04 3
05 .
06 1
07 4
08 1
09 5
10 9
11 2
12 6
13 5
14 4
15 x
```

The *-strip* option can be used to remove leading characters from source. E.g. if someone sent you the above as area.txt and you tried to compile it you would get the following error:

```
ERROR: illegal instruction: 01 ENTER, line: 1
```

However if you strip off the first 4 characters all is well:

```
vcomp.pl comp -strip 4 -new 12c -nst 12c.nst area.txt
```

```
Phase 1: Preprocessor
Phase 2: Check GTOS
Phase 3: Compile
Done!
```

Macros

Macros are defined in vcomp.pl. To add or edit use your favorite text editor and search for "model Macro Start", e.g. "12C Macro Start". This is an example of two 12C macros:

```
12C Macro Start
&SMOD
LBL START
-
LSTx
x<=y
GTO START
RV
&MOD
x<>y
```

```
ENTER
RV
x<>y
/
LSTx
x<>y
INTG
*
x<>y
-
CHS
12C Macro End
```

Macro definitions start with &NAME and end when the next macro starts and may contain any number of legitimate statements and virtual labels (12C only). E.g. the &SMOD (slow or subtract mod) leverages the use of 12C virtual labels to setup a loop.

In this example two 15C macros are defined. One for calling the other as a subroutine:

```
&CALL_MOD
GSB .8
&MOD
LBL .8
x<>y
ENTER
RV
x<>y
/
LSTx
x<>y
INT
*
R^
-
CHS
RTN
```

An example of how to use this macro:

```
LBL A
. . .
&CALL_MOD
. . .
&CALL_MOD
. . .
RTN ;End of program
&MOD
```

- NOTE:** Macros cannot call macros.
- NOTE:** Macros do not support numeric expansion.

Compiler Preprocessor

Phase 1 of the compiler checks for syntax, expands macros, checks for duplicate labels, processes virtual labels, and expands multiple digit numbers. If there is an error the compiler will halt and display a useful message. E.g. the following 12C program has an invalid statement on line 3:

```
11
&SMOD
SIN
```

The compiler will report:

```
ERROR: illegal instruction: SIN, line: 3
```

Errors in macros will be reported as:

```
ERROR: illegal instruction: SIN, in macro: &TAN, line: 2.1
```

The 2.1 indicates the 2nd line of your code and the 1st line of the macro.

Phase 2 of the compiler checks for valid GTO statements and completes the processing of the virtual label GTOs. If a GTO has nowhere to go an error message will be produced and the compiler will halt. E.g., the following 12C program has an invalid GTO:

```
11
&SMOD
GTO A
```

The compiler will report:

```
ERROR: Virtual GTO A goes nowhere, post processed line: 8
```

Line 8? The source only has 3 lines. Finding the source of this error may be difficult with larger programs. This can be addressed with the *-phase* option. *-phase 1* will halt the compile and dump out intermediate results, e.g.

```
vcomp.pl comp -phase 1 -new 12c -nst 12c.nst foo.txt
```

Phase 1: Preprocessor

```
001    1f 1
002    1f 1
003    b2 -
004    d5 LSTx
005    c5 x<=y
006    ---- gtostart
007    dc Rv
008    ---- gtoa
```

The first column is the line number. Column 2 is the byte code. The gtoa on line 8 is the offending statement. The ---- statements require the next compiler phase, to see the byte code and the final GTO statements use *-phase 2* after replacing GTO A with RTN, e.g.:

```
vcomp.pl comp -phase 2 -new 12c -nst 12c.nst foo.txt
```

Phase 1: Preprocessor
Phase 2: Check GTOs

```
001    1f 1
002    1f 1
003    b2 -
004    d5 LSTx
005    c5 x<=y
006    03 GTO 03
007    dc Rv
008    00 GTO 00
```

For long programs the *-c* and *-cc* options can be used with *-phase* to columnize the output.

Authors

vcomp.pl was written by Thomas Klemm and Egan Ford.

Test with Nonpareil

To test your program with Nonpareil:

1. Shutdown Nonpareil. Important! Nonpareil on exit writes out its state.
2. Compile your program as a new state file.
3. Replace the Nonpareil state file with your new state file.
4. Startup Nonpareil.

This process can be simplified if you compile to an existing Nonpareil state file. After closing Nonpareil Windows users would type:

```
vcomp.pl comp -nst "c:\Documents and Settings\Egan\nonpareil\15C.nst" code.ext
```

Linux/UNIX users:

```
vcomp.pl comp -nst ~/.nonpareil/15C.nst code.txt
```

I prefer this method because all my register and stack variables are still intact.

Accelerate your Nonpareil

Nonpareil will execute your programs at authentic Voyager speeds. It can be very frustrating to have a competition or challenge code take 9 hours just to return the wrong answer. If you have the ability to build your own version of Nonpareil you can speed up program execution by about 200 times.

First edit src/model.c and change the following lines from:

```
{ "11C", PLATFORM_VOYAGER, ARCH_NUT, 40, 215000 },
{ "12C", PLATFORM_VOYAGER, ARCH_NUT, 40, 215000 },
{ "15C", PLATFORM_VOYAGER, ARCH_NUT, 80, 215000 },
{ "16C", PLATFORM_VOYAGER, ARCH_NUT, 40, 215000 },
```

to:

```
{ "11C", PLATFORM_VOYAGER, ARCH_NUT, 40, 2147483647 },
{ "12C", PLATFORM_VOYAGER, ARCH_NUT, 40, 2147483647 },
{ "15C", PLATFORM_VOYAGER, ARCH_NUT, 80, 2147483647 },
{ "16C", PLATFORM_VOYAGER, ARCH_NUT, 40, 2147483647 },
```

This will increase the emulated frequency to as fast as possible. Don't get too excited, you are still limited by the speed of your computer.

Next edit src/voyager_lcd.c, and change:

```
#define VOYAGER_DISPLAY_BLINK_DIVISOR 150
```

to:

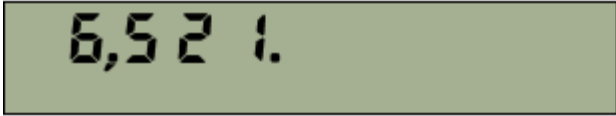
```
#define VOYAGER_DISPLAY_BLINK_DIVISOR 0
```

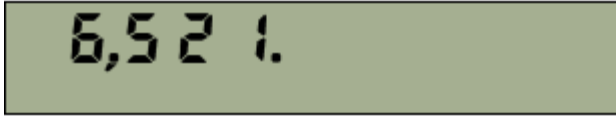
This will disable the blinking "running". This will not increase the speed of the emulation, but will reduce the OS overhead measurably when running accelerated. This is important for the next step to get the most out of your pimped-out Voyager.

Lastly edit src/proc.c and fiddle with:

```
#define MAX_INST_BURST 5000
```

If you make this number too big you will be unable to use the virtual keyboard. This number will vary from system to system. Increase this value by increments of 5000 until Nonpareil becomes unstable (keyboard does not work) or CPU utilization reaches 100% when running your program. For my 2.4 GHz P4 system I have MAX_INST_BURST set to 30000. My CPU utilization is 98%. A program that took 9:37 hours with Nonpareil and a real 15C now runs in 3:05 minutes.

Runtime: 0:09:37:33

Date: Tue Mar 18 06:19:47 UTC 2008

Runtime: 0:00:03:05

Date: Mon Mar 17 17:55:34 UTC 2008

It is probably not a good idea to replace your normal Nonpareil. To avoid overwriting when compiling with scon.py do not specify the install option.

Linux example:

To build/test:

```
cd nonpareil-0.78
scons - local/scons.py
build/posix/nonpareil 15c
```

When you have a satisfactory supercharged Nonpareil binary, just copy it to /usr/local/bin as nonpareil.fast, e.g.:

```
cp ./build/posix/nonpareil /usr/local/bin/nonpareil.fast
```

Test the Real Thing

Unfortunately there is no way to electronically transfer your programs from the virtual to the physical world. But you can use the listing features of the compiler to provide the necessary keystrokes to save time and increase accuracy.

Errata

Reported bug in vcomp 15C listings has been fixed.

Edited: 30 Oct 2012, 9:19 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Eramco Systems RSU-2 Soft Select (128K) for HP-41

Posted by [Diego Diaz](#) on 1 Apr 2008, 12:41 a.m.

Introduction

This document is intended to provide a useful guide to configure ERAMCO RSU-2 unit in order to gain access to its 128K (32 pages).

There is a well documented ERMACO OS that fits this unit conveniently but such a document seems almost impossible to find. To overcome the lack of the original handling software documentation as well as for those that preferred to use his/her RSU-2 without the original OS, I've designed a few steps procedure in order to allow RSU-2 users managing the "huge" amount of memory in their RAM-Boxes. Anyhow if someone is the fortunate proprietary of an original RSU-2 OS it would be of great help to all of us if he or she decided to share the original documentation of this software with the HP-41 enthusiasts community.

What follows has been deduced by reverse-engineering my own RSU-2 and has also been tested in the RSU-2 borrowed to Emmanuel Compès. Thanks to him and to the superb job of schematics reverse engineering and compilation done by Tony Duell.

Technical details

First let's have a look at what those guys at ERAMCO did back in the eighties to handle 128K with an HP-41. They were really ingenious indeed.

As you all know pages #0 to #7 of an HP-41 are reserved for internal usage. Thus there shouldn't be any RAM in these addresses. With this in mind writing (using the H'040 instruction) into any address lower than H'8000 would be harmless.

A "register" were placed in this area (below H'8000) to keep the configuration word of the RSU-2. The format of said register as well as the register's access method is explained in the following paragraphs.

The 128K of the RSU-2 are organised into four "Banks" containing 32K each. Every "Bank" holds pages from #8 to #F in an all-in-a-row fashion (i.e. no Bankswitching allowed).

The "Banks" into RSU-2 have nothing to do with "Banks" as referred in the Bank-switching schemes.

How to configure

There is an eight bits "register" containing required configuration bits inside the RSU-2, I've found a way of writing into that "register", thus allowing full control over said configuration and over RSU's 128K.

You can reach the configuration register by "writing" (H'040) into **any** address in the range H'0000 to H'7FFF (lower half of the 64K addressable space)

The address you choose to write at is meaningless, the word (10 bits) you write is where attention must be paid:

- bit 9 - So far I've found no use of it.
- bit 8 - Set(1) or clear (0) the addressed bit in the register
- bit 7 - So far I've found no use of it.
- bit 6 - So far I've found no use of it.
- bit 5 - So far I've found no use of it.
- bit 4 - So far I've found no use of it.
- bit 3 - So far I've found no use of it.
- bit 2 - Address bit 2 of the register's addressed bit
- bit 1 - Address bit 1 of the register's addressed bit
- bit 0 - Address bit 0 of the register's addressed bit

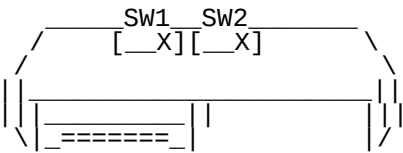
With this pattern, if you want to set (i.e.) bit 5 of the configuration register, just write H'105 into any address below H'8000 (I used H'4100)

Now the meaning of those register bits is needed to get the whole picture, here you are:

- Bits 7-6 = address of current active Bank (00=0, 01=, 10=2, 11=3) each Bank holds 32K
- Bit 5 = Enable/disable page #D
- Bit 4 = Enable/disable page #C
- Bit 3 = Enable/disable page #B
- Bit 2 = Enable/disable page #A
- Bit 1 = Enable/disable page #9
- Bit 0 = Enable/disable page #8

Pages #E & #F cannot be disabled thru configuration register.

Pay attention to the function of the two switches you'll find in the top (or front) of the unit.



SW1 SW2
ON OFF ON OFF (viewing with the connector towards you)

SW1: ON=Write protect pages #E & #F, OFF=Write enabled.

SW2: ON=Read enable pages #8 to #F, OFF=Read disabled.

IMPORTANT: Please note that you **must** make sure that addresses of polling points (from H'XFF4 to H'XFFA) are cleared for every page and plane prior to set SW2 ON (Read enable position) otherwise severe-crashes are most likely to occur. Also make sure that the internal lithium battery is in good working order to prevent your data from accidental erasing when the RSU-2 is unplugged from the HP-41.

Configuration procedure and register table

According to the info detailed above you'll need a means to write into your HP-41's memory to properly configure RSU-2. I've chosen myself the HEXEDIT command from HEPAX module as it also allows a convenient CLRAM command which can easily erase the random (garbage) contents of the RSU-2 setting it into an initialized (zeroed) status. To avoid HEPAX RAM interferences with RSU-2 RAM I've used a Clonix-41 module pre-configured as ROM-only HEPAX and fixed to page #7.

Any other software allowing WRITE s&x (H'040) instruction can be used (e.g. David Assembler) or, if you're comfortable with M-code, you can build your own routines for that specifis tasks. Just remember thah you'll have to

allocate it into a page number out of the range #8 to #F as these are going to be used by RSU-2.

First you must set SW2 to OFF position to avoid garbage reading and consequent crash of your HP-41. Also set SW1 to OFF so you can write pages #E and #F. Then insert the Clonix into any port (1 to 3).

Now plug your RSU-2 into the HP-41, turn it on and execute HEXEDIT.

XEQ (ALPHA) HEXEDIT (ALPHA)

You'll see [ADR: _ _ _ _] on the display.

NOTE: You may want to assign HEXEDIT and CLRAM commands to a couple of keys for your convenience.

Choose any address below H'8000 (I use H'4100) and write in sequence:

H'007, H'006 (This will set "Plane 0" as the "active" plane) H'100, H'101, H'102, H'103, H'104, H'105 (This will enable pages #8 to #D)

Exit HEXEDIT by double clicking the back arrow [<-] key.

Write "OK" into ALPHA, this is required for the CLRAM command to operate)

Now we need to erase the whole 32K (8 pages) of active plane:

1) 8 ENTER (Place 8 into HP-41's X register.)

2) Execute CLRAM.

Repeat steps 1) and 2) for values 9 to 15 (pages #9 to #F)

Now execute HEXEDIT again and write H'106 (This will switch plane 1 to active)

Exit HEXEDIT and repeat erasing procedure from 8 to 15

Enter HEXEDIT once more and write H'107, H'006 to activate plane 2.

Exit HEXEDIT and repeat erasing procedure from 8 to 15

Execute HEXEDIT for the last time and write H'106 to activate plane 3

Again, clear pages #8 thru #F as you've done previously.

After finishing this (certainly annoying) procedure we've got a shinny initialized 128K RSU-2 unit ready to play with. Fortunately this should only be done once. Nevertheless you cannot read its contents until you set SW2 back to ON position. Go ahead and do it now.

Here you are two tables summarizing the words you must write to handle the whole thing.

	PAGE #8	PAGE #9	PAGE #A	PAGE #B	PAGE #C	PAGE #D
To enable	H'100	H'101	H'102	H'103	H'104	H'105
To disable	H'000	H'001	H'002	H'003	H'004	H'005

Table 1: Codes to enable/disable “read” on individual pages. Note that you can still write into a disabled page, but the data won’t be readable untill you re-enable the corresponding page.

	BANK 0	BANK 1	BANK 2	BANK 3
To activate	H'007,H'006	H'007,H'106	H'107,H'006	H'107,H'106

Table 2: Codes to activate one of the four "Banks".

Handling RSU-2 with its own OS

For those that have the original RSU-2 OS but (as most of us) lack its documentation, I've prepared this short reference as a guide to the three new undocumented functions.

The rest of the functions are fully covered by the "Eramco ES-41 Database 85" manual.

Port numbers in these functions are referred to in the following manner:

```
Physical Port 1: Page # 8 => X = 1
Physical Port 1: Page # 9 => X = -1
Physical Port 2: Page # A => X = 2
Physical Port 2: Page # B => X = -2
Physical Port 3: Page # C => X = 3
Physical Port 3: Page # D => X = -3
```

Note that pages #E and #F are not handled by these functions, they should contain a copy of the "ESRSU-1A" and "ES RSU-2A" respectively, in each of the four Banks, to work properly.

None of the following functions is programmable.

- **PORTON (XROM 04, 62)**

Input: Port number in X

Output: (none)

Fuction: Allows access to the port specified by the contents of X register.

- **PORTOFF (XROM 04, 63)**

Input: Port number in X

Output: (none)

Fuction: Disables access to the port specified by the contents of X register.

Note: After PORTOFF is executed, the selected page can not be read, but it can still be written with WRITE s&x (H'040) M-code instruction.

- **SETBANK (XROM 06, 56)**

Input: Bank number in X (Bank number must be in the range 1 to 4)

Output: (none)

Fuction: Selects the Bank specified by the contents of X register as the active Bank.

Note that the configuration of the enabled/disabled pages remains unchanged when you switch from one active Bank to another.

Also the Bank and pages configuration, as well as the RAM contents, are kept by the internal battery while the RSU-2 is disconnected from the HP-41.

Please experiment by yourselves and do not hesitate in mailing me your questions or comments.

Best wishes and enjoy your RSU-2... ;-))

Diego Díaz.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Making sense of all the 41c Service ROM versions: 1A 1C 1D 2A 2B

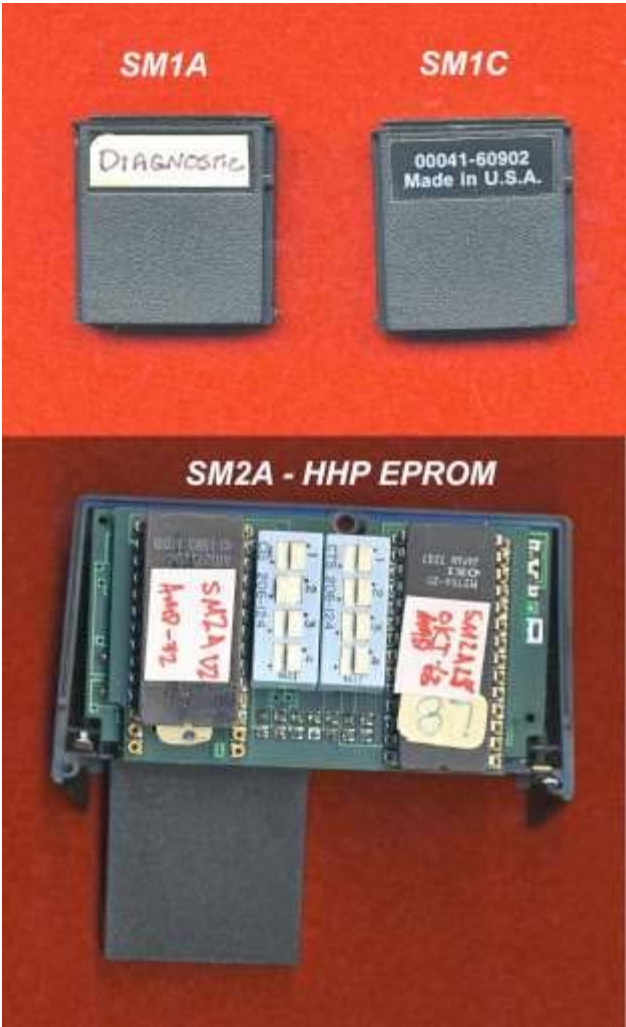
Posted by [Allen](#) on 10 Mar 2008, 12:04 a.m.

[Making sense of all the 41c Service ROM versions: 1A 1C 1D 2A 2B](#)

Confused about the HP 41C Diagnostic/Service Roms? Chances are, you're not alone. There are five known types of HP 41C service ROM, and few concise guides exist regarding their use and nomenclature. Although they are all very rare, the two most common versions (SM1C and SM2A) are both called "Diagnostic" or "Service" module interchangeably- This can make using and trading them very confusing!

Here I discuss the key mapping and distinct advantages the of four different Service ROMS: (will be edited to add the 1D Version if time allows)

1. SM-1A prototype version of SM-1C?
2. SM-1C (a.k.a. ET-11966)
3. SM-1D To be resolved at a later date
4. SM-2A (a.k.a. 5061-7221)
5. SM-2B 82143A Printer and ROM tester (not shown)



A [PDF Version](#) of the Key Map is available

Key Code	Func .	none none none SM1A	HP 00041 - 60902 ET-11966 SERVICE .MOD SM1C	none 5061 - 7221 SERVICE - CX .MOD SM2A	none ET-11968 82143A SERVICE ROM SM2B
01	ON	ON	ON	ON	ON
02	USER	ALL TESTS	ALL TESTS	CX - MULT - TEST	PRINTER TEST
03	PRGM	(AUTO)	(AUTO)	CV - MULT - TEST	(AUTO)
04	ALPHA	(AUTO)	(AUTO)	(AUTO)	(AUTO)
11	SIGMA	KEYBOARD	KEYBOARD	CX - KEYBOARD	N/U
12	1/X	KEY - CODE	KEY - CODE	CX - KEY - CODE	N/U
13	SQRT	DISPLAY	DISPLAY	CX - DISPLAY	N/U
14	LOG	DATA STORAGE	DATA STORAGE	CX - DATA STORAGE	N/U
15	LN	ROM TEST	ROM TEST	CX - ROM TEST	N/U
21	x<>y	KEYBOARD	KEYBOARD	CV - KEYBOARD	N/U
22	R DN	KEY - CODE	KEY - CODE	CV - KEY - CODE	N/U
23	SIN	DISPLAY	DISPLAY	CV - DISPLAY	N/U
24	COS	DATA STORAGE	DATA STORAGE	CV - DATA STORAGE	N/U
25	TAN	ROM TEST	ROM TEST	CV - ROM TEST	N/U
31	SHIFT	STANDBY	STANDBY	STANDBY	N/U
32	XEQ	SLEEP	SLEEP	SLEEP	N/U
33	STO	CHARACTER	CHARACTER	CHARACTER	N/U
34	RCL	MEM - MOD	MEM - MOD	MEM - MOD	N/U
35	SST	APP - MOD1	APP - MOD1	4K APP MOD	N/U
41	ENTER	APPL - MOD ID	APPL - MOD ID	APPL - MOD ID	N/U
42	CHS	CPU	CPU	CPU	N/U
43	EEX	I/O PORT	I/O PORT	X - MEM MOD	N/U
44	< - -	APP MOD 2	APP MOD 2	8KAP ROM	N/U
51	-	ROM ID	ROM ID	ROM ID	N/U
52	7	N/U	N/U	TIME CHIP (CX)	N/U
53	8	N/U	N/U	GEN ROM TST	GEN ROM TEST
54	9	SERV MOD TST	SERV MOD TST	SERV MOD TST	SERV MOD TST
61	+	N/U	CR QA	N/U	N/U
62	4	N/U	CR MOTOR	TIME MOD (CV)	N/U

63	5	N/U	CR BUFFER	X FUNCT (CV)	N/U
64	6	N/U	CR READ TST	12K APP MOD	N/U
71	*	N/U	CR WRITE TST	N/U	N/U
72	1	N/U	N/U	FREQ (W/TIME)	N/U
73	2	N/U	N/U	N/U	STATUS REG
74	3	N/U	N/U	16K APP MOD	EOL TEST
81	/	DEBOUNCE	DEBOUNCE	DEBOUNCE	N/U
82	0	N/U	N/U	N/U	N/U
83	.	N/U	N/U	N/U	N/U
84	R/S	STOP TEST	STOP TEST	STOP TEST	STOP TEST

N/U =NOT USED

SM 4K page must be loaded on PG 4 of HP41 memory

SM1A Advantages

1. More Rare than SM1C modules
2. Can test MEM (82106A) and 4k/8k APP modules

SM1C Advantages

1. Adds Card reader tests to the SM1A
2. Can test MEM (82106A) and 4k/8k APP modules

SM2A Advantages

1. Adds 12k, 16K, X-FUNCTION, X-MEMORY ROM tests
2. Adds TIME chip, TIME MOD, and FREQ test
3. SM2A can test QUAD mem rather than 1MEM
4. Disadvantage: NO card reader test available in SM2A

SM2B Advantages

1. Can test 82143A Printer functions as well as General ROM test

A note regarding the emulation of Service ROMs:

ROM emulators can be a cheap and easy way to emulate SERVICE ROMs. For example many of the SM2A MODULE tests require the module to be in PORT 4.

There are a few exceptions where the modules will work in port 2:

1. GENERAL APP
2. QUAD MEM
3. TIME MOD
4. X MEM

References/Further Information

1. SM1A- No further information available
2. SM1C- See **HP-41C Service Manual** available [here](#)
3. SM2A- See **HP-41C Service Manual** available [here](#)
4. SM2B- See **HP 82143A Service Manual** available elsewhere online

Edited: 10 Mar 2008, 12:33 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

41C to 50g Migration

Posted by [Egan Ford](#) on 1 Mar 2008, 4:48 a.m.

Abstract

Busted, lost, stolen, etc... Words that describe the fate of many beloved 41Cs. However, life after the 41C is possible. This article describes how to run your favorite 41C programs on your relatively cheap, replaceable, and handsome 50g.

NOTE: the following article also applies to the 48 and 49 series.

41C Emulators for the 50g

HrastProgrammer has developed a complete line of 41C emulators that include support for a number of popular plug-in modules. These emulators are available for the 48/49 series and the 50g. This article focuses on the freely available HP-41E. The commercial version ([HP-41X](#)) offers enhanced features not available with HP-41E.

HP-41E is an HP-41CV emulator that comes in eight different flavors. This article is based on the HP-41EA flavor. A complete description of each flavor can be obtained here: <http://hrastprogrammer.com/hp41e/index.htm>.

Installing HP-41E

The follow instructions assume that you have an SD card with at least 350 KB free.

1. Power off your 50g and remove the SD card.
2. Insert the SD card in to your desktop/laptop SD card reader.
3. Create an HP41E directory.
4. Download <http://hrastprogrammer.com/hp41e/hp41e50.zip> and extract into the HP41E directory on the SD card.
5. Put the SD card back into the 50g.
6. Create an HP41E folder in the 50g HOME directory.
7. Create an HP41 program in the HP41E folder with the following contents:

```
<< 3: "HP41E/HP41EA" EVAL >>
```

Alternatively you can just drop HP41EA into any 50g directory and run it from there. This will save you the hassle of creating a RM41 file, but it will take up to 75 KB of RAM, whereas RM41 requires less than 7 KB of memory.

If you are just browsing and want to take a test drive first, then fire up Debug4x (EMU48 with 50g support), drag-and-drop HP41EA onto the virtual LCD (first level stack element should be 'Code'), then type 'HP41' ENTER STO.

Running HP-41E

From your 50g HOME/HP41E directory run the newly created HP41 program. You'll be greeted with a MEMORY LOST message:



Try not to get too excited, we still have a bit of work to get done. Ok, go ahead and add a few numbers together, get it out of your system. When you are done playing around use ON-F to exit. Now run HP41 again. Doh! MEMORY LOST message again. No problem, you just need to enable memory saving. To do this press ON-STO. Now add a few numbers together, then ON-F. You'll notice a new file in your 50g HOME/HP41E directory: RM41. This is the persistent HP-41E storage. If you remove that file you lose your memory.

If you want to keep your RM41 file on SD, first copy it to SD:

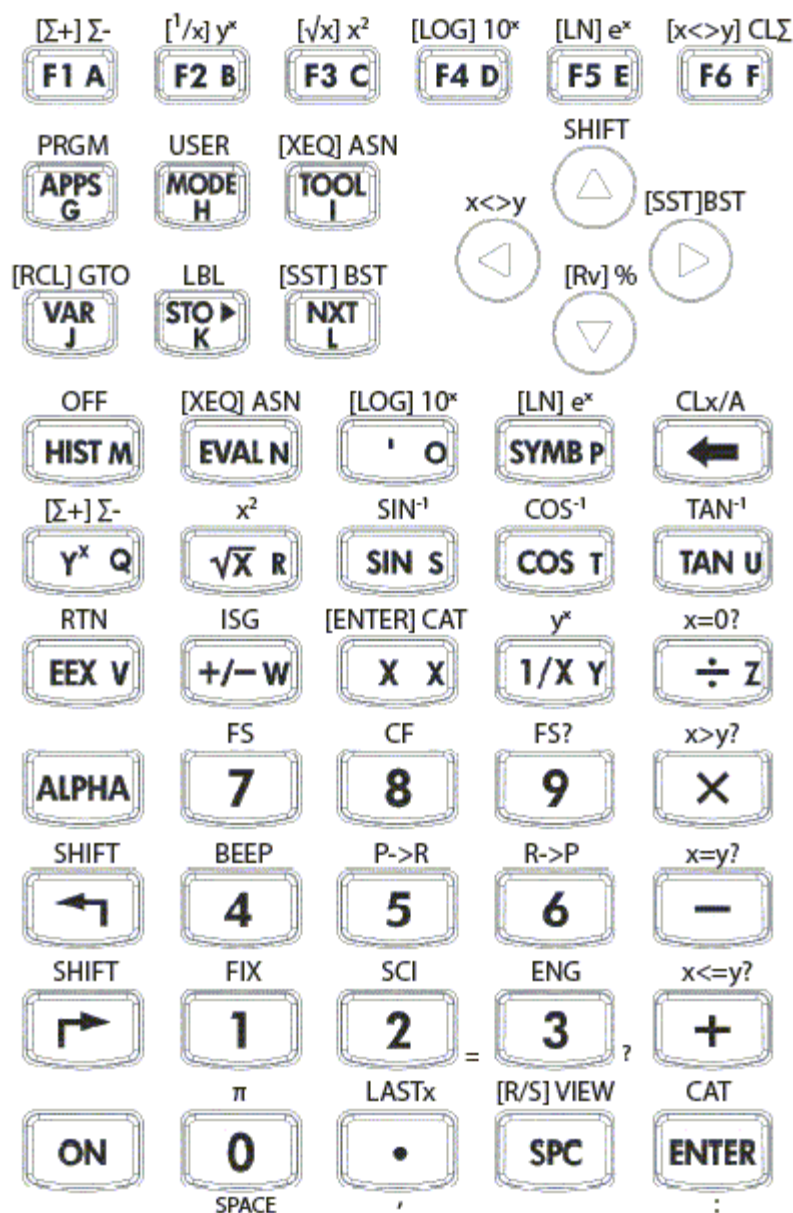
```
'RM41' RCL
:3:"HP41E/RM41" STO
'RM41' PURGE
```

then change the HP41 launch program to:

```
%%HP: T(3)A(R)F(.);
\<<
:3:"HP41E/RM41"
RCL 'RM41' STO
:3:"HP41E/HP41EA" EVAL
'RM41' DUP RCL
:3:"HP41E/RM41"
DUP PURGE
STO PURGE
\>>
```

This is also useful for desktop to 50g shuttle transfers.

As you experiment with HP-41E, you'll notice that there is not a 41C to 50g key for key equivalent. Until you memorize the differences, print out this keymap, laminate it, and put in your wallet:



The labels in brackets ([]) are the replacement for the label on the key. The unbracketed labels are the shifted function.

Programming HP-41E

Programming HP-41E is no different than a real HP-41CV except that its harder until you memorize the keypad. Alternately there is an easier way. Using a couple of freely available tools you can compile your 41C listing in to a RAW file and then use the RAW41 utility to insert them into your RM41 file.

Big picture process:

1. Create a working directory on your desktop/laptop. E.g. C:\WKDIR.
2. Copy RM41 from your 50g and place in working directory.
3. Place 41C programs as text files in working directory.
4. Use HP41UC to compile text source files into RAW files.
5. Use RAW41 to insert RAW programs into RM41 file.
6. Copy RM41 to 50g.

RAW41 can be obtained from: <http://hrastprogrammer.com/hp41e/raw41.zip>, and HP41UC from: <http://www.hpmuseum.org/software/41uc.htm>.

TIP: I store my RM41 file on my SD card in the HP41E directory. HP41UC, RAW41, and my source files are in that directory as well. When I need to manage my RM41 file, I just put the SD card in my laptop and change to the SD HP41E directory and do all my work there. Makes it easy.

NOTE: HP41UC is a DOS program and can be run from any DOS emulator (e.g. DOSBox). However RAW41 is a Windows executable. For RAW41, Mac/Linux users will need Windows installed in a virtual machine. I recommend VMware Fusion for the Mac and VMware Workstation 6.5 for Linux.

Wine (Windows Emulator) is also an option. An example Wine session from a Linux workstation:

```
$ wine RAW41.EXE \? RM41
```

```
HP-41E Program Manager
Copyright (C) 1998-2000 by HrastProgrammer

LOAD RM41 ...
CALC=HPHP49-C
REGS=320 SIZE=273 R00=0EF BOTTOM=040 TOP=3EF
FIRST=0C0 .END.=0C8.2 PRGM=0D4.6 LINE=FFF
PROGRAMS:
  0C8.2 C80C20 .END.
  0D4.6 C600F7 LBL "SORTEM"
  0D5.2 C80500 END
  0DA.6 C600F7 LBL "SAVAGE"
  0DB.2 C80800 END
  0E3.6 C600F2 LBL "S"
  0E4.2 C80900 END
  0ED.6 C600F5 LBL "NQ41"
  0EE.2 C00000 END
```

A few things to consider with Wine and Linux or OS/X:

1. Names are case aware AND case sensitive (Windows is just case aware).
2. ? is a meta character, to safely pass ? to RAW41.EXE, use \?. If you do not you run the risk of ? being expanded to the first filename in your working directory that is one character long. Granted its low risk, but still a risk.

Wine can be obtained from: <http://www.winehq.org/>.

Compiling Programs with HP41UC

For example create the program COOL.41C with your favorite text editor (Word is NOT a text editor, try notepad):

```
;my cool program
LBL "COOL"      ;always comment your code
"HP-41E IS"     ;put string in alpha register
>" SO COOL"     ;put more string in alpha register
AVIEW           ;show the string
END             ;end of program
```

Compile with HP41UC:

```
C:\WKDIR>DEL COOL.RAW
C:\WKDIR>HP41UC /t=COOL.41C /r=COOL.RAW
```

Output:

```
txt[ C:\WKDIR\COOL.41C ] => raw[ C:\WKDIR\COOL.RAW ]
.END. found on line 6.
type[ E080 ]
size[ 0020 ] ( 32 bytes )
implementation[ 00200020 ]
256 bytes written.
```

For more information on HP41UC read the documentation: <http://www.hpmuseum.org/software/41uc.htm>.

Inserting Programs into HP-41E Memory

First list programs:

```
C:\WKDIR>./RAW41 ? RM41
```

Output:

```
HP-41E Program Manager
Copyright (C) 1998-2000 by HrastProgrammer
LOAD RM41 ...
CALC=HPPHP49-W
REGS=320 SIZE=273 R00=0EF BOTTOM=040 TOP=3EF
FIRST=0C0 .END.=0EE.2 PRGM=0EF.0 LINE=000
PROGRAMS:
    0EE.2 C00020 .END.
```

Empty. No programs. Load program:

```
C:\WKDIR>RAW41 COOL.RAW RM41
```

Output:

```
HP-41E Program Manager
Copyright (C) 1998-2000 by HrastProgrammer
LOAD RM41 ...
RAW=COOL.RAW CALC=HPPHP49-W
REGS=320 SIZE=273 R00=0EF BOTTOM=040 TOP=3EF
FIRST=0C0 .END.=0EE.2 PRGM=0EF.0 LINE=000
PROGRAMS:
    0EE.2 C00020 .END.
LOADING FROM RM41 TO RM41 (CHECKSUM=D8 FREE=322 RAW=32) ...
CALC=HPPHP49-W
REGS=320 SIZE=273 R00=0EF BOTTOM=040 TOP=3EF
FIRST=0C0 .END.=0E9.2 PRGM=0ED.6 LINE=FFF
PROGRAMS:
    0E9.2 C80420 .END.
    0ED.6 C600F5 LBL "COOL"
    0EE.2 C00000 END
```

Cool program loaded. You can verify again with:

```
C:\WKDIR>RAW41 ? RM41
```

Output:

```
HP-41E Program Manager
Copyright (C) 1998-2000 by HrastProgrammer
LOAD RM41 ...
CALC=HPPHP49-W
REGS=320 SIZE=273 R00=0EF BOTTOM=040 TOP=3EF
FIRST=0C0 .END.=0E9.2 PRGM=0ED.6 LINE=FFF
PROGRAMS:
    0E9.2 C80420 .END.
    0ED.6 C600F5 LBL "COOL"
    0EE.2 C00000 END
```

For more details on RAW41 read the documentation: <http://hrastprogrammer.com/hp41e/raw41.htm>.

Running HP-41E Programs

After you have copied RM41 from your desktop/laptop back to your 50g, then run HP41, and type:

```
EVAL ALPHA C O O L ALPHA
```

Easy!

Summary

There is an abundance of 41C programs available. It is easy to download, compile, and load into HP-41E without actually having to type in the programs. As for your own programs, well at least you can type them in your desktop/laptop first, compile, then load.

Your 41C programs live on. Enjoy!

References

- <http://hrastprogrammer.com/hp41e/index.htm>
- <http://hrastprogrammer.com/hp41e/raw41.htm>
- <http://www.hpmuseum.org/software/41uc.htm>

Edited: 10 Apr 2008, 4:00 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

HP-41C RECURSION SOLVES IT

Posted by [designnut](#) on 25 Feb 2008, 3:54 p.m.

THE SEALED VOLUME Cubic supplied a telemetry transmitter to be injected into orbit that had to operate on the ascent as well as in orbit. The high power section only was to be enclosed in a pressurized sealed chamber to prevent arc-over due to the RF power level. The lower power stages were not pressurized. A hermetic coaxial connector seal and mating surface was ordered to an 80-microinch finish. Inspection checked the surface for roughness, and then scribed crossed lines to check the centering. That did wonders for the finish. It was flushed with a mixture of helium and nitrogen, in a glove box, and then sealed. After sealing with an O ring, it was necessary to check the leak rate as it was to hold pressure for 24 months. The initial leak rate seemed much higher than the published seal leak rate. Leakage from the whole package was read. Data was available every 5 minutes for 4 hours, where the leakage seemed to stabilize at the rate the seal manufacturer predicted. The customer thought the low leak rate might mean the gas had all leaked out, and needed proof. Cubic and Lockheed had been faced with this problem for 2 years. CURVE PLOTTING Everyone had the same data. My boss, Blair plotted data on linear graphs 25 feet long, in an attempt to explain the behavior. I bought a new HP 41C pocket calculator with curve fitting capability. I keyed in the data, assuming an exponential leak rate, the correlation was 0.384, definitely not exponential. I retried it using a power law curve, and got a 0.996 correlation, it was a power law curve. The leakage rate was in very small numbers, like 10^{-8} to 10^{-11} cc/second. People had used time scales in minutes, hours, days and months to graph the data. I chose seconds to plot the time scale as the leakage rate was in cc/sec. I used a log-log plot to cover data from 5 minutes to 2 years, and the long leakage scale. Data fell on a straight line and after 4 hours became asymptotic to a constant leak rate predicted by the seal manufacturer. I knew a power law curve plotted as a straight line on log-log paper. I have always been expert at graphing. THE VOLUME LINE Since I had a rate in cc/seconds and a seconds time scale I realized the product was the volume of the enclosed space. I got an estimate of 50cc for that volume and plotted a sloping volume line at 50cc. There was a surprising piece of data that had been run on the equipment with the unit processed but no seal installed; none was available. I plotted the measured leakage rate after two months, it was almost off the graph, but when the initial line was extended it ran right through this point. Now I knew what was happening, it was all the parts that had adsorbed helium, and were outgassing. The customer was able to supply some long term data at 18 and 24 months. When plotted, it was on a transitional curve from the predicted seal leak rate, to the volume line. When back tracked, you could see a slight fall in the leakage rate as it neared the volume line. An exponential curve on a log-log plot would show a slight drop before the time constant point. It dropped 10% at 0.1 of the time constant. The data all fit. The leak history was nowhere near the volume line, until the long term data points, the gas could not have leaked out. RESULTS! I showed Blair and the customer the results, nobody cared. When asked, Blair thought someone else had made the plot. Nothing succeeds like success. Their nose was bent out of shape as I solved a 2 year old problem that neither company had solved. It took me a few hours to solve this knotty problem.

Edit Message

Delete Message

Password:

[\[Return to the Message Index \]](#)

[Go back to the main exhibit hall](#)





HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

Control the World with HP-IL/RS-232/41CX + X10

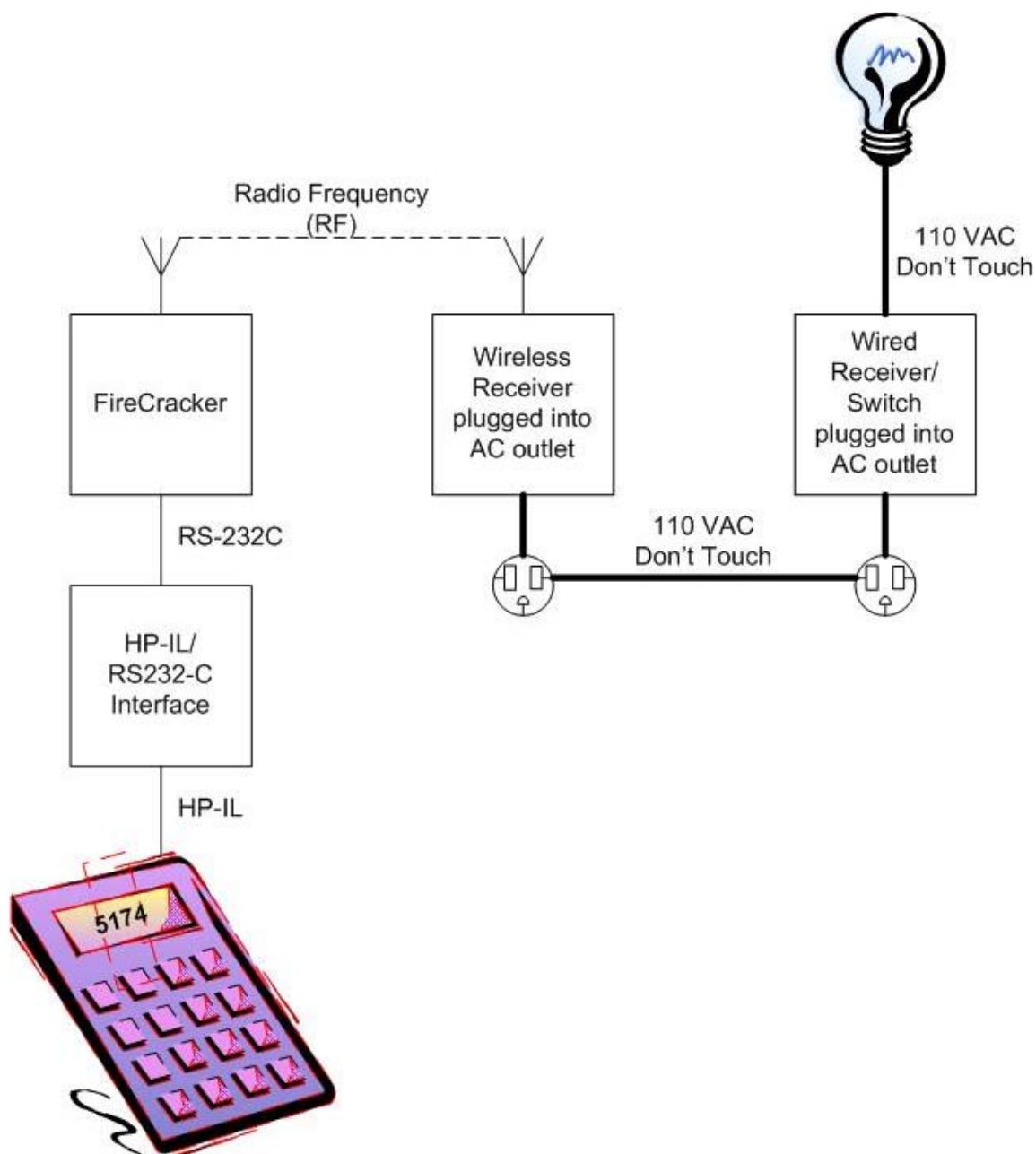
Posted by [Egan Ford](#) on 8 Sept 2007, 4:04 p.m.

Abstract

This article describes how to do *light* home automation with an HP 41CX, an HP-IL/RS-232C interface, and the X10 FireCracker.

X10 FireCracker

The FireCracker Computer Interface sends Wireless Radio Frequency (RF) signals from your PC (or 41CX) to an X10 Transceiver. The Transceiver then passes the signals onto your house wiring to control lights or appliances around your home. You plug lights and appliances into X10 Modules.[1]



The FireCracker is a small RS-232 powered DB9 dongle. On one end is a DB9 male connector and on the other a DB9 female connector. The FireCracker was designed to be used as a pass-through device. I.e. you can still use your PC or HP-IL/RS-232 serial port for another device. This is accomplished by inserting the FireCracker between the host and the other device.



Because the FireCracker supports pass-through programming is not as straightforward as programming other serial output devices (e.g. printers). Each FireCracker/X10 command is a string of 40 bits, but instead of sending 5 bytes down the traditional data transmit line of the RS-232C interface, we toggle the Data Terminal Ready (DTR) and Request To Send (RTS) lines to send 0's and 1's. Fortunately the HP-IL/RS-232 interface supports this level of control.

The FireCracker derives its power supply from either the RTS or DTR signals from the serial port. At least one of these signals must

be high at all times to ensure that power is not lost from the FireCracker.[2]

Signal	RTS	DTR
Reset	0	0
Logical '1'	1	0
Logical '0'	0	1
Standby	1	1

To send a 0 or 1 lower then raise the appropriate line, never lower both. E.g. to send 101, first lower RTS and DTR to reset the FireCracker, then raise DTR and RTS to standby mode. To send the first 1 lower DTR, then raise DTR. To send the 0, lower RTS, and then raise RTS. To send the last 1 lower DTR, then raise DTR.

Every FireCracker/X10 command is 40 bits long. The specification can be obtained from [2]. IANS, the first 16 bits is a static header, the next 16 is our data, and the last 8 is a static footer. The data bits are created by two values, the address of the device to control and the action to perform.

Addresses are labeled A1-A16 - P1-16 for a total of 256 devices that can be controlled. Actions are limited to on, off, bright, dim. There is no way to query the current state of the target device. But you can always change the state to exactly what you want. E.g. power on, send 5 dim commands, etc... If you power off, then back on, the dim is forgotten. There is no way other than visual inspection (or some other type of sensor) to determine if an action actually took place. I.e. there is no feedback or error notification. Think about something else for mission critical applications (e.g. your home made automatic pet feeder).

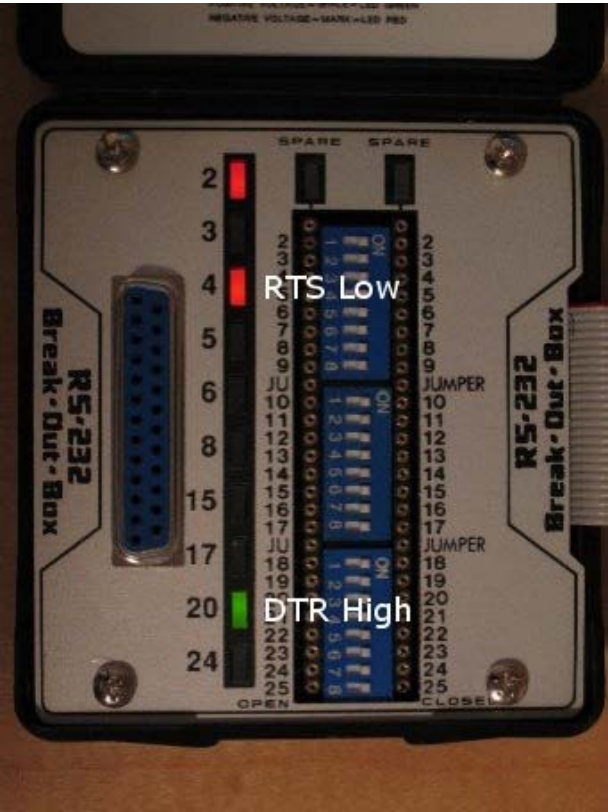
Generating the 16 data bits is nontrivial; evidence of this is in the length of the specification. 85% of the specification is the data bit patterns. There is no logical correlation between bit position, addresses, or actions. There is some order, but there could be more. A table will need to be created of 32 values once and saved for future use. See [2] for more detail.

41CX to HP-IL/RS-232C to X10 Firecracker

Compared to the 71B/GPS article [3] this is a breeze. There are only 4 commands that we need to use:

LI1	DTR	High
LI2	DTR	Low
LI3	RTS	High
LI4	RTS	Low

E.g. connecting a breakout box in-between the FireCracker and the HP-IL/RS232 interface and sending LI1;LI4 (the ; is a command separator required by the HP-IL/RS232 interface) yields:

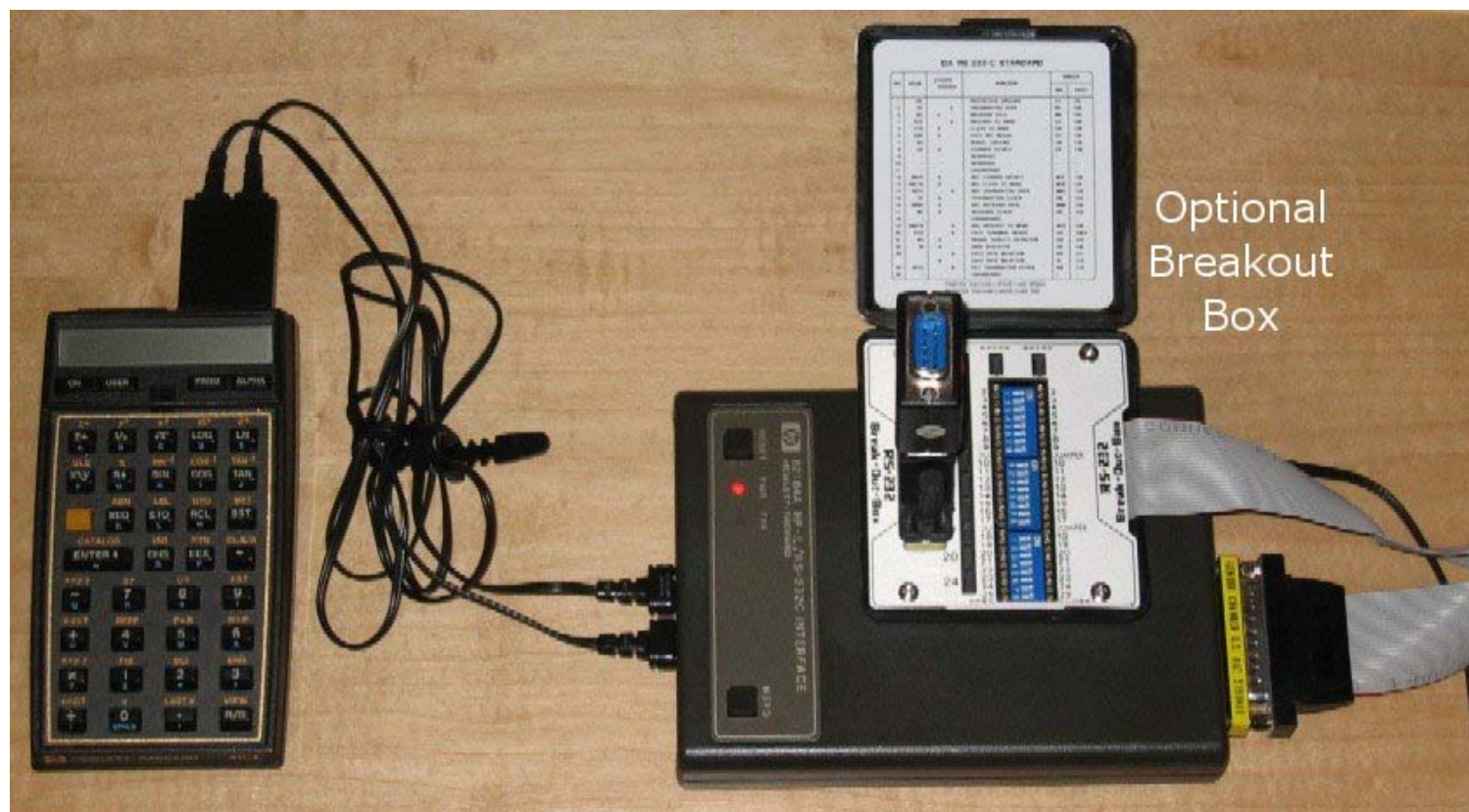


Equipment Setup

Use the following diagrams to assemble your 41CX home automation system. NOTE: 41CX Extended I/O Module not required.



Optional breakout box setup:



Source Code

A zip file with text, raw, and barcode (pdf) can be obtained here: <http://www.hpmuseum.org/guest/eganford/x10.zip>.

;X10

```
;Protocol: http://software.x10.com/pub/manuals/cm17a_protocol.txt
;
;Place device address in ALPHA register, e.g. E2
;Place a value -19 to 20 in the X register
;X = -19 to -1 = number of times to click dim
;X = 1 to 19 = number of times to click bright
;X = 0 = off
;X = 20 = on
;
LBL "X10"
ASTO 32 ;STO X10 address (A1-A16 - P1-P16)
STO 33 ;STO action (0=OFF, 20=ON, 1-19=BRIGHT, -(1-19)=DIM)
SF 25 ;ignore next error
"X10.DAT" ;file containing address bits
FLSIZE ;size of file? to check if file exists
FC?C 25 ;if error (file not found) create file, clear ignore next error flag
XEQ 30 ;create X10.DAT with address bits
"X10.DAT" ;file containing address bits
0 ;set pointer to register 0
SEEKPTA
000.031 ;load up registers 0-31 from X10.DAT
GETRX
AUTOIO ;initialize firecracker
"HP82164A"
FINDIO ;FINDID (XROM 28,28), hp41uc defines it as FINDIO?
SELECT
REMOTE
"LI2\x3bLI4" ;set DTR/RTS low, reset mode, using \x3b for ; because of hp41uc use of ; for comments
OUTA
"LI1\x3bLI3" ;set DTR/RTS high, stanby mode, using \x3b for ; because of hp41uc use of ; for comments
OUTA
CLA ;clear alpha register
ARCL 32 ;restore x10 address
ATOX ;get ASCII code
65 ;subtract 65 to get A-P as 0-15
-
RCL IND X ;rcl x10 A-P address bits
4096 ;left shift 12 bits
*
STO 34 ;sto in 34

20
RCL 33 ;off?
X=0?
GTO 10
X=Y? ;on?
GTO 11
X<0? ;dim?
GTO 12
GTO 13

LBL 10 ;off bits
32
STO+ 34

LBL 11 ;on bits + 2nd part of address from ALPHA register
ANUM
15
+
RCL IND X
STO+ 34
0
STO 35 ;sto count
GTO 20

LBL 12 ;dim- bits
CHS
16
STO+ 34
RDN

LBL 13 ;dim+ bits
136
STO+ 34
RDN
STO 35 ;sto count
GTO 20

LBL 20 ;build 40 bits
213
STO 45 ;header byte 1
170
STO 44 ;header byte 2
173
```

```
STO 41          ;footer byte
RCL 34
256
/
INT
STO 43          ;data byte 1
RCL 34
256
MOD
STO 42          ;data byte 2

LBL 35          ;count loop for dim/bright
5
STO 40
LBL 21          ;loop for 5 bytes/40 bits
RCL 40
40
+
RCL IND X
X<>F
8
STO 39
LBL 22          ;loop for 8 bits
RCL 39
1
-
STO 38
FS? IND 38
XEQ 23
FC? IND 38
XEQ 24
DSE 39
GTO 22
DSE 40
GTO 21
DSE 35
GTO 35

LOCAL          ;No REMOTE
0              ;clear flags
X<>F
CLA           ;clear ALPHA
ARCL 32       ;restore X10 address
RCL 33        ;restore X register arg
RTN           ;END

LBL 23        ;send 1 bit
"LI2"         ;set DTR low
OUTA
"LI1"         ;set DTR high
OUTA
RTN

LBL 24        ;send 0 bit
"LI4"         ;set RTS low
OUTA
"LI3"         ;set RTS high
OUTA
RTN

LBL 30        ;load up X10.DAT
34
CRFLD
6
STO 00
7
STO 01
4
STO 02
5
STO 03
8
STO 04
9
STO 05
10
STO 06
11
STO 07
14
STO 08
15
STO 09
12
STO 10
```

```
13
STO 11
0
STO 12
1
STO 13
2
STO 14
3
STO 15
0
STO 16
16
STO 17
8
STO 18
24
STO 19
64
STO 20
80
STO 21
72
STO 22
88
STO 23
1024
STO 24
1040
STO 25
1032
STO 26
1048
STO 27
1088
STO 28
1104
STO 29
1096
STO 30
1112
STO 31
0
SEEKPTA
000.031
SAVERX
RTN
END
```

Usage

Enter the address of the device in the ALPHA register as A1-A16 - P1-A16, e.g. E2. Then enter into the X register a number between -19 and 20, where 20 is on, 0 is off and any other number is the number of times you want to brighten or dim. The documentation indicates that each bright and dim request changes the light by 5%, in practice that is not true. I'd say it's closer to 10%.

After entering the values for device address and action, XEQ X10 then wait ~40 seconds. Each bit takes about 1 second to send. If you are trying to impress someone XEQ X10 first, then start to explain how it works. NOTE: The registered EMU41 with HP-IL adapter in fast mode takes a few seconds.

After execution the ALPHA and X registers will be restored to their initial state so that a follow up action can be entered quickly.

41CX as a Glorified Light Timer

So far all that I have described is an expensive remote for the FireCracker/X10 home automation system. BTW, you get a remote as part of the system. But, can the free remote wake up and turn on and off the lights? No! But your 41CX can!

1. Make sure your 41CX clock is set correctly.
2. Write a small program for lgton and lgtoff:

```
LBL "LGTON"
"E2"          ;address of device
20            ;20 = power on
X10           ;X10 program
RTN           ;end
```

```
LBL "LGTOFF"
"E2"          ;address of device
```

```
0           ;0 = power off
X10         ;X10 program
RTN         ;end
```

3. Create an on and off alarm:

```
"^^LGTON"    ALPHA SHIFT ENTER SHIFT ENTER LGTON ALPHA
0
0
18.00        Turn on lights at 6 PM.
XYZALM       XEQ ALPHA XYZALM ALPHA

"^^LGTOFF"   ALPHA SHIFT ENTER SHIFT ENTER LGTOFF ALPHA
0
0
23.00        Turn off lights at 11 PM.
XYZALM       XEQ ALPHA XYZALM ALPHA
```

Impress Your Date (or not)

Using the timer functions of the 41CX you can easily set the mood without being too obvious.

1. Plan a dinner date; setup the lights to start dimming 10% every 15-30 minutes.
2. DON'T show your date your 41CX setup--ever. Keep that between us on this forum.
3. If your date catches on that the lights are dimming, lie. Say that you paid an embarrassing high sum for auto dimming; scratch that--nerdy, say mood lighting. That will impress her. Actually if you eBay'd your equipment it's not a lie.
4. Resist the urge to show off your contraption. If you elect to come clean and convince her to visit your basement (dungeon) or home office (tower), she'll just scream at the site of your franken-calculator-remote-control and run out the door. Dumbfounded you still have the sense to use your franken-calc to turn on the lights as she runs out the door in the dark, but the 40 second delay is too slow. If she does not break her leg on the way out she will return hours later with an angry mob bearing torches (why do they fear us?).

Development Environment

My development environment consists of a 200 Mhz Pentium Overdrive-based Red Hat 9 Linux workstation with 32MB of RAM and an HP 82973A HP-IL ISA card. I use the registered version of EMU41[4]+DOSEMU to emulate an HP-IL equipped 41CX. Exploiting Linux allows me remote access from my notebook to develop and test 41CX/HP-IL applications quickly and easily. After development is complete it takes only a few seconds via HP-IL to transfer the application to a physical 41CX for final verification. This workstation can also extend the functionality of a physical 41CX by emulating an HP-IL display, printer, and floppy.

Unlike the 71B where BASIC editing is very quick and easy, 41CX/EMU41 RPN entry and editing can be laborious. Fortunately there are options. I use a standard text editor to write all my 41CX code including comments. The code can then be "compiled" by the wonderful HP41UC[5] utility, then loaded in to EMU41 for testing. E.g.:

1. Create the program cool.rpn:

```
;my cool program
LBL "COOL"           ;always comment your code
"41CX IS SOO COOL"   ;put string in alpha register
AVIEW                ;show the string
END                  ;end of program
```

2. Compile with HP41UC:

```
del cool.lif
hp41uc /t=cool.rpn /l=HDRV2.DAT
```

3. Edit EMU41.INI file to include HDRIVE2, e.g.:

```
[DEVICES]
DISPLAY
HDRV1
HDRV2 HDRIVE2.DAT
LPRTER1
DOSLINK
XIL 1700
```


4. Load up program for testing in EMU41:

```
"HDRIVE2"      F4 HDRIVE2 F4
FINDID         F8 F4 FINDID F4
```

The number 3 should be returned to the stack if HDRIVE2 is the 3rd device listed in EMU41.INI

```
SELECT         F8 F4 SELECT F4
"C00L"         F4 C00L F4
READP          F8 F4 READP F4
1
SELECT         F8 F4 SELECT F4
```

If you connect your 41CX to the loop the instructions are exactly the same. Easy!

Summary

Using the 41CX for home automation would have been a hit in the 80s. Today it is of no practical use other than a learning aid and some mild entertainment.

References

- 1. <http://software.x10.com/pub/manuals/ck18a-is.pdf>
- 2. http://software.x10.com/pub/manuals/cm17a_protocol.txt
- 3. <http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/articles.cgi?read=746>
- 4. <http://membres.lycos.fr/jeffcalc/emu41.html>
- 5. <http://www.hpmuseum.org/software/41uc.htm>

Edited: 25 June 2008, 12:27 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous\]](#) | [Next](#)

Explore the World with HP-IL/RS-232/71B + GPS

Posted by [Egan Ford](#) on 8 Sept 2007, 3:32 p.m.

Abstract

This article describes how to connect an NMEA compliant RS-232 GPS receiver to a 71B to find your position and to set your clock. This article does not describe why you would want to do that.

GPS, NMEA, and RS-232

The National Marine Electronics Association (NMEA) defined a RS-232 communication standard for devices that include GPS receivers. The GPS receivers can output geospatial location, time, headings and navigation-relevant information in the form of ASCII comma-delimited message strings[1].

Example NMEA output:

```
$GPRMC,163814,A,5331.6289,N,11331.8077,W,0.0000,0.000,050907,,*35
$GPGSA,A,3,04,23,01,31,20,25,02,16,13,,,1.6,0.8,0.7*3E
$GPRME,0.00,M,0.00,M,1.54,M*1E
$GPGSV,3,1,12,04,16,291,36,23,80,285,42,06,10,034,00,01,08,108,33*74
$GPGSV,3,2,12,31,23,064,39,20,41,192,41,25,25,244,41,02,11,327,43*7C
$GPGSV,3,3,12,16,51,105,38,27,07,244,33,13,47,291,46,138,28,171,34*4F
$GPGGA,163815,5331.6292,N,11331.8077,W,2,09,0.80,704.4,M,-19.783,M,,*46
```

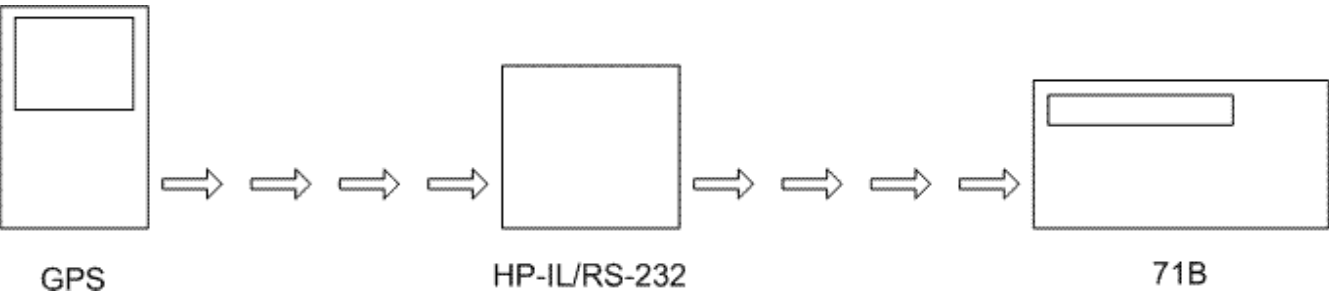
For the purposes of this article we are only interested in the line that starts with \$GPGGA. \$GPGGA is our fix--our position at some time and is formatted:

```
$GPGGA,UTC Time,Latitude,N/S,Longitude,E/W,,,,Elevation,Units,,,,*Checksum
```

This GPS continuously streams this data out the GPS RS-232 interface. To capture this into the 71B for processing we need an RS-232 interface.

GPS to 71B via HP-IL/RS-232

The HP 82164A is an HP-IL to RS-232 interface. This interface allows HP-IL-based hosts such as the 71B to communicate with RS-232-based devices.



The above animation illustrates a perfect world scenario. The data flows effortlessly from the GPS to the 71B. However, in our imperfect world some understanding of flow control and coordination is required to make it appear as if it was a perfect world.

The following is a crude simplistic generalization that better illustrates what really takes places as data streams from the GPS to the 71B.



1. The 71B makes a request for a line of data terminated with a linefeed (LF). The 71B can request data as lines or bytes. This is accomplished using the 71B ENTER statement. Plain text lines are obviously easier to work with and fortunately that is what NMEA has specified. When the ENTER command has received a complete line it will instruct the HP-IL talker (the HP-IL/RS-232 interface in this case) to stop sending.
2. With communications halted the 71B can process data obtained from ENTER without loss of additional data.
3. The HP-IL halt has a ripple effect. The HP-IL/RS-232 interface will need to capture in its 109 byte receive buffer the GPS stream to prevent data loss while the 71B processes the current data set.
4. The HP-IL RS-232 receive buffer has a user definable threshold that is less than the size of its 109 byte receive buffer. When the threshold is exceeded a halt transmission is sent to the GPS. The threshold (default 24 bytes) setting depends on how fast the RS-232 transmitter (GPS) responds to a halt transmission command (XOFF in this case). E.g. my GPS is slow to respond, so I had to reduce this threshold to 12 bytes. The buffer needs room to continue to capture data while the halt transmission command is being sent and processed. If the threshold is too large we run the risk of an overrun and data loss. Slower GPS devices will need a smaller threshold.
5. When the GPS is instructed to halt transmitting data it starts to fill up it's transmit buffer. Not shown in this animation is what happens when the GPS buffer fills up. The GPS cannot request that the satellites stop transmitting. The cached data must be discarded to make room for new data. The non-stop flow of data is a challenge with collecting and processing GPS output. Most data communications require that no data be lost, flow control can halt traffic for long periods of time while the host processes data. Often lost data can be requested and resent. This is not the case with GPS, the data is not cached, it is temporal and has less value the older it is.
6. Fortunately the circa 1985 71B can process the data faster than the data can be collected and sent by the GPS. When the 71B requests the next line of data, the HP-IL/RS-232 interface flushes the buffered data.
7. When the HP-IL/RS-232 receive buffer falls below the threshold it issues a start sending message (XON in this case) to the GPS. The GPS rapidly flushes its buffer and the cycle starts all over again with no data loss.

This works for three important reasons:

1. The GPS collects and produces data slower than its RS-232 transmit rate of 4800 bit-per-second. This allows buffers to be flushed without loss while collecting more data.
2. The 71B is fast enough to process the received data before the GPS buffer fill up. We are partially responsibility for this by keeping our data processing logic small. If our code was too complex it may take longer to run than the GPS buffer to fill up and data will always be lost. In situations like this it may be better to not process the data, but just capture it into the relatively large buffers of 71B RAM and then process later. The data will not be as fresh, but is

still as accurate since GPS transaction are time stamped using a time source independent of the 71B. E.g. say you wanted to track movement. The logic to process movement data real-time could be time consuming, it may be better to collect all the data, and then when movement has stopped process the data.

3. Buffers. They exist in all asynchronous digital communication devices. Without them as programmers we would have write very tight time sensitive code or have a way to request lost data. The temporal nature of GPS receivers prevents this.

Equipment Setup

Use the following diagram to assemble your 71B GPS data receiver.



Obviously not something to pack on your next camping trip. Of course when smash your GPS receiver screen, you'll wish you had a way to obtain a fix.

Source Code

Lines 10-20: Initialization. D0 (0 or 1), if set to 1 every line of text received from the GPS will be displayed. This can slow down the processing of the data, leading to buffer overruns. U (-12 to 12) is your UTC offset. D (0 or 1), DST (Day Light Savings). A (0 or 1), display elevation, and E\$ (M or FT) is the elevation units.

```
10 DESTROY ALL @ OPTION BASE 1 @ STD
20 D0=0 @ U=-7 @ D=1 @ A=1 @ E$="FT" @ DIM G$[100]
```

Lines 30-70: HP82164A initialization. The most important settings are in bold. The 12 is the receiver buffer block size in bytes (24 is the default). When 12 bytes have been received, the GPS is sent an XOFF and does not resume until the buffer has been cleared by the 71B. If you experience corrupted data decrease this number. SBC sets the 4800 BPS rate, and LI5 disables DTR control. If LI5 is not set the data will be duplicated (unexpected buggy behavior). FYI, the HP82164A default settings are 9600 BPS 8N1 XON/XOFF.

```
30 X=DEVADDR("HP82164A")
40 SEND UNT UNL LISTEN X MTA
50 SEND DDL 2 DATA 13,10,13,10,17,19,5,6,106,17,12,17
60 SEND UNT UNL
70 REMOTE @ OUTPUT :HP82164A ;"SBC;LI5" @ LOCAL
```

Lines 80-140: Data collection loop. Lines terminated with LF are collected one at a time, if they do not start with \$GPGGA, then get the next line, else verify that the checksum is a match, if the match fails get the next line. Keep this loop tight to avoid buffer overruns.

```
80 ENTER :HP82164A ;G$
90 IF D0=1 THEN DELAY 0,0 @ DISP G$
100 IF G$[1,6]="$GPGGA" THEN 120
110 GOTO 80
120 K=LEN(G$)-2
130 IF G$[K,K]<>"*" THEN 80
140 CALL CHECKSUM(G$,K$) @ IF G$[K+1,K+2]<>K$ THEN 80
```

Lines 150-190 displays the time and synchronizes the 71B clock to the GPS clock (accurate within a second). If there is no valid fix it jumps back to the data collection loop.

```
150 G$=G$[8,LEN(G$)]
160 IF G$[1,1]=" " THEN DISP "TRACKING SATELLITES" @ WAIT 5 @ GOTO 80
170 H=VAL(G$[1,2])+U+D @ IF H<0 THEN H=H+24
180 M=VAL(G$[3,4]) @ S=VAL(G$[5,6])
190 CALL SYNCTIME(H,M,S)
```

Lines 200-270 extracts and displays your position.

```
200 L0=VAL(G$[8,9]) @ L1=VAL(G$[10,11]) @ L2=VAL(G$[12,16])*60
210 L3=VAL(G$[20,22]) @ L4=VAL(G$[23,24]) @ L5=VAL(G$[25,29])*60
220 DISP USING "#,T: ",ZZ," ",ZZ," ",ZZ," ",L: " ";H;M;S
230 DISP USING "#,K, ",K";L0;L1 @ DISP " ";STR$(L2);" ";G$[18,18];" ";
240 DISP USING "#,K, ",K";L3;L4 @ DISP " ";STR$(L5);" ";G$[31,31];
250 IF A=1 THEN CALL ALTITUDE(G$,E$)
260 DISP
270 END
```

Calculate checksum subroutine. XOR the numeric value of each character between the \$ and the * and return a 2 digit hex checksum.

```
280 SUB CHECKSUM(S$,K$)
290 K0=NUM(S$[2,2])
300 FOR I=3 TO LEN(S$)-3 @ K0=BINEOR(K0,NUM(S$[I,I])) @ NEXT I
310 K$=DTH$(K0)[4,5]
320 END SUB
```

Synchronize time subroutines. This routine builds a SETTIME compliant string and calls SETTIME.

```
330 SUB SYNCTIME(H,M,S)
340 T$=""
350 CALL PAD(H,N$) @ T$=N$&" ":
360 CALL PAD(M,N$) @ T$=T$&N$&" ":
370 CALL PAD(S,N$) @ T$=T$&N$
380 SETTIME T$
390 END SUB
400 SUB PAD(N,N$)
410 N$=STR$(N) @ IF N<10 THEN N$="0"&N$
420 END SUB
```

Elevation subroutine.

```
430 SUB ALTITUDE(S$,E$)
440 C0=0 @ A0=0 @ A1=0
```

```
450 FOR I=1 TO LEN(S$)
460 IF S$[I,I]="," THEN C0=C0+1
470 IF C0=8 AND A0=0 THEN A0=I+1
480 IF C0=9 AND A1=0 THEN A1=I-1
490 NEXT I
500 A=VAL(S$[A0,A1])
510 IF S$[A1+2,A1+2]="M" AND E$="FT" THEN A=INT(A*3.281+.5)
520 DISP " A:";A;E$;
530 END SUB
```

Example Output (D0 = 1)

```
$GPRMC,163814,A,5331.6289,N,11331.8077,W,0.0000,0.000,050907,,*35
$GPGSA,A,3,04,23,01,31,20,25,02,16,13,,1.6,0.8,0.7*3E
$PGRME,0.00,M,0.00,M,1.54,M*1E
$GPGSV,3,1,12,04,16,291,36,23,80,285,42,06,10,034,00,01,08,108,33*74
$GPGSV,3,2,12,31,23,064,39,20,41,192,41,25,25,244,41,02,11,327,43*7C
$GPGSV,3,3,12,16,51,105,38,27,07,244,33,13,47,291,46,138,28,171,34*4F
$GPGGA,163815,5331.6292,N,11331.8077,W,2,09,0.80,704.4,M,-19.783,M,,*46
```

```
T: 10:38:15 L: 53 31'37.752" N 113 31'48.462" W A: 2311 FT
```

Before you come knocking on my door or try looking in my backyard from Google Earth[2] please note that this is not output from my GPS, but output that I cut and pasted from a GPSd server in Canada. I am sure they would love a visit from you. I like visitors too--the invited kind.

Testing

As you can imagine dragging a 71B + HP-IL/RS-232 w/ power + GPS outside may not be practical. Fortunately for me, my GPS works indoors. However, if you are not so lucky and need to test GPS software, you can get real data from a number of freely accessible GPSd[3] servers on the Internet--Google for gpsd (e.g. <http://gpsd.mainframe.cx/>).

Once you have obtained a set of data (10 or so lines should be fine), configure your PC serial terminal software (e.g. HyperTerminal) with 4800 BPS, 8N1, XON/XOFF, and terminate lines with LF (Hint: HyperTerminal/Properties/Settings/ASCII Setup/Send line ends with line feeds). Then cut/paste the output.

Development Environment

My development environment consists of a 200 Mhz Pentium Overdrive-based Red Hat 9 Linux workstation with 32MB of RAM and an HP 82973A HP-IL ISA card. I use the registered version of EMU71[4]+DOSEMU to emulate an HP-IL equipped 71B. Exploiting Linux allows me remote access from my notebook to develop and test 71B/HP-IL applications quickly and easily. After development is complete it takes only a few seconds via HP-IL to transfer the application to a physical 71B for final verification and timing. This workstation can also extend the functionality of a physical 71B by emulating an HP-IL display, printer, and floppy.

Summary

Using the 71B to process GPS data would have been a hit in 1985. Today it is of no practical use other than a learning aid, setting your 71B clock, and some mild entertainment.

References

- 1. <http://www.boondog.com/tutorials/gps/gps.html>
- 2. <http://earth.google.com>
- 3. <http://gpsd.berlios.de>
- 4. <http://membres.lycos.fr/jeffcalc/emu71.html>

Edited: 9 Sept 2007, 12:34 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-35s bug list

Posted by [Paul Dale](#) on 22 Aug 2007, 6:22 p.m.

Here is a quick list I've made up which contains the list of known bugs in the HP-35s. Feel free to contact me with new bugs and expansions for these bugs.

1. Vector input bug has been mentioned and very weird behavior shows up but I'm not aware of this being isolated or repeatable. It might be related to memory full. I do have enough trust in the people reporting problems to believe there is something to them.
2. Cos(x) for x near 90 is dud. For what appears to be the same reason, Tan(x) for x near 90 is also dud.
3. Checksums are meaningless. Seems to be related to numbers in programs but nothing definitely proven yet.
4. Program sizes are also meaningless. Again seems to be number related and again nothing proven.
5. Bad error message on indirection on J: 1000 ; STO J ; x<>(J). You get "INVALID (I)" error message if done from the keyboard, from a program it gives the correct message.
6. Bad radix. If you are in RADIX, mode and you enter RADIX, in program mode it is displayed as "RADIX.". If you run the program it behaves correctly.
7. Program entry bug for large programs. Create a 999 step program and then try to enter a new LBL. You're not allowed. Delete one step, create the LBL and put the step back and all is fine.
8. In a program: VIEW (I) ; PSE will show the wrong value of I.
9. In ALL mode, the mantissa should be properly rounded so that the exponent fits on the screen. i.e. the entire number is displayed without scrolling.
10. This sequence as a program displays (37): 10 instead of (10): 10. LBL Z ; 10 ; STO I ; STO (I) ; VIEW I ; PSE It works fine in run mode and without the PSE. I've had a report that the keystrokes here should be: LBL Z ; 10 ; STO I ; STO (I) ; VIEW (I) ; PSE. Can somebody confirm please.
11. Pressing and releasing the binary operator keys real fast results in missed strokes.
12. The '1 (f) /c' sequence to display the current denominator doesn't work in ALG mode but is fine in RPN mode.
13. TANH, SINH and COSH all have bugs when applied to large values. An OVERFLOW message given when computing TANH for values greater than or equal to 30000. The TANH value is correct, 1, but the OVERFLOW message is wrong. SINH and COSH generate the OVERFLOW message correctly, but return the wrong values. Overflow is supposed to return 1E500, but with an argument greater than or equal to 30000 SINH and COSH return the mantissa of the argument and the characteristic is changed to 499. e.g. SINH 31415 = OVERFLOW --> 3.1415E499


```

14. LBL A
    156.25
    STO X
    208.33333334 ;There are eight 'threes' in there
    STO R
    1.77951304201
    STO Q
    -R*X/(X*Q-R) ;Should evaluate to roughly -467, and it does
    -R*X/(X*Q-R) ;Should (still) evaluate to roughly -467, but
    calculator outputs 31.323 instead!
    RTN

```

15. Some programs cannot be interrupted. Prompt messages without a PSE following them cause the problem:

```

A001 LBL A
A002 SF 10
A003 LOCKUP DEMO
A004 0
A005 GTO A004

```

Warning: running this code will mostly likely cause all memory to be lost.

16. When scrolling through 'line count' program memory [via the <-MEM 2] menu, downward, I will find a LBL ; that the list seems to be stuck at. Seems to be related to have unlabeled statement(s) at the start of program memory and entering this mode when at the top of program memory (GTO ..)

17. Input $-1/3i - 1/3$ by typing ". 1 . 3 CHS i . 1 . 3 CHS ENTER". Get "0.3333i-0.3333" instead of "-0.3333i-0.3333". Works correctly when entering "0.3333333 CHS i 0.3333333 CHS ENTER".

18. It appears that registers G...L are used internally in the 2*2 lin. solve in an unusual way. If there is anything other than a real number in these registers (a complex number or a vector) the linear solver returns "no solution." Kind of interesting, as these registers are not overwritten after using the linear solver, which appears to indicate that these registers are not actually used.

19. Deleteing a line with a label on it, doesn't correctly update branches to that label and in fact can leave you with a branch to a non-existent step. The documentation says it should update.

K001 LBL K	Deleting	K001 LBL K	
K002 INPUT A	L001 gives	K002 INPUT A	
K003 XEQ L001	----->	K003 XEQ L001	this should be XEQ K006
K004 VIEW B		K004 VIEW B	
K005 RTN		K005 RTN	
L001 LBL L		K006 B=pi*SQ(A)	
L002 B=pi*SQ(A)		K007 RTN	
L003 RTN			

20. Solve infinite loops for some program based functions. Assign FN=D and SOLVE the following for X. The solve doesn't finish. This seems to be related to using a loop in the function being solved.

```

D001 LBL D
D002 10
D003 STO N
D004 DSE N
D005 GTO D004
D006 RCL X
D007 COS
D008 RTN

```

A workaround for this is to include an always false conditional after any INPUT statements with a RTN instruction and a STOP. For example:

```

B001 LBL B
B002 INPUT T
B003 INPUT X
B004 CF 1
B005 FS? 1
B006 RTN

```

```
B007 FS? 1
B008 XEQ C001
B009 0.2xX-T
B010 RTN
C001 LBL C
C002 STOP
C003 RTN
```

Now for some items that cannot be classified as bugs since they are documented or work correctly. These items are still quite annoying and should be fixed:

1. GTO.a ENTER doesn't work as expected. GTO.a000 does.
2. INPUT (i) for $i \geq 0$ doesn't work. This is documented in the manual but still seems wrong.
3. Poor choice for the graphics for the "theta". It is much too close to the graphic for "8".
4. No easy way to extract to deconstruct complex numbers into pairs of reals.
5. No rectangular to polar conversions except for display purposes.
6. 2×2 lin. solve and 3×3 lin. solve look strange. 2×2 and 3×3 seems better. i.e. multiplication sign instead of asterisk.
7. Input of numbers in bases other than 10 require the suffix to be explicitly entered. This makes non-base 10 unworkable.

- Pauli

Edited: 23 Sept 2012, 8:55 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

UserRPL stack commands

Posted by [James M. Prange \(Michigan\)](#) on 8 July 2007, 3:51 a.m.

It has been asserted that the RPL stack manipulation commands are "numerous and unintuitive".

Numerous? Well, perhaps, but most of the UserRPL stack commands can be grouped into just three "families", plus three more commands that don't really fit into those families. Unintuitive? I think not, at least for most of them.

With the fixed-depth 4-register stack of "Classic RPN", the stack commands are few and simple. With the added flexibility of RPL's variable-depth stack, having more (and more flexible) stack commands seems desirable. If you really wanted to, you could get by with just the stack commands DEPTH, either ROLL or ROLLN, PICK, and DROPN (along with some extra keystrokes or programming).

With Classic RPN models, the T register is replicated downward as arguments are taken from the bottom of the stack, so an endless supply of whatever is in the T register is available as an argument for additional operations. With RPL's variable-depth stack, this feature isn't available. Perhaps the simplest work-around for the loss of this feature is to store an object in a named variable (local or global), and then calling the variable by name can provide an endless supply of the object to level 1, for as long as the variable exists unchanged. If you need the object to be on a different level, then call it by name and use stack manipulation to move it to wherever you need it. Alternatively, DEPTH PICK will copy the topmost level down to level 1, which has the potential to provide an endless supply of the object in the topmost level to level 1 (as long as the topmost level isn't changed). Of course, if you need the object to be on a different level, then follow DEPTH PICK with a stack manipulation to move it to wherever you need it, or perhaps store it in a variable for later use.

Regarding the RPL model "series", I treat the 28C and 28S as the "28 series", the 48SX, 48S, 48GX, 48G, and 48G+ as the "48 series", and the 49G, 49g+, 48gII, and 50g as the "49 series".

Regarding character translations in this article, "<<" and ">>" represent the UserRPL program delimiter characters, and "\->" represents the right-arrow character.

For the "stack diagrams", of course there can be an indefinite number of levels above those that I show, in which case, except for being moved up or down, those higher levels are unaffected.

Regarding LAST, or alternatively LASTARG on the 48 and 49 series, the assumption is that this is enabled, that is, on the 28 series, flag 31 is set, and on the 48 and 49 series, flag -55 is clear.

Where I show sequences or programs that have the same effect as a command, that refers only to the stack itself, not to what LAST or LASTARG will return.

I really don't think that the UserRPL stack commands are difficult to learn, although keeping track of exactly which objects are on which levels can be difficult when you use more than a few levels. Well, you can always use named global or local variables when it seems too difficult to keep track of the stack.

Of course, being careful of the order in which objects are put on the stack in the first place helps to reduce the need for stack manipulation; it may not be needed at all.

If you're going to need an object again, then use a stack manipulation that copies it, and for its final use, use a stack manipulation that moves it instead.

In general, it helps to have a "clean" stack when you start something new. Unneeded objects on the stack use up memory and may slow down execution, and of course add "clutter". Discard (or store in a global variable for future use) anything that you don't need on the stack anymore, either just before starting something new, or else when you finish what you're doing.

Of course, to manipulate a stack level, that level has to actually exist, or else you'll get an error. But some of these commands take a number for an argument, and if the argument is 0, then these (except for UNPICK and NDUPN, each of which requires two arguments) don't error out even with an empty stack.

For stack commands that take a numeric argument, non-integer values are rounded to integer values, and a negative number is treated the same as 0. With the 49 series, either a zint (an "exact integer") or a "real number" can be used. The commands that return a count to the stack always return it as a real number.

Also note that the commands that take a numeric argument first remove the numeric argument from the stack, and then act on the remainder of the stack, so when you use n as the numeric argument, it refers to the stack levels as they will be after n is removed from the stack.

Note that the stack doesn't really have any objects in it. That you seem to see objects in the stack is due to a bit of indirection. The stack is really a stack of 5-nibble pointers (that you don't see) with the objects pointed to elsewhere in memory. When the "stack display" needs to be updated, the objects pointed to are decompiled to string forms suitable for display with the current stack display modes and font size, and those are what you see. The stack manipulation commands can be fast because objects of various types and sizes aren't being manipulated; only the fixed-size stack pointers are being manipulated. For example, when you use DUP to make a "copy of the object on level 1", you aren't really making a copy of the object, you're just making a copy of the pointer, and when you use SWAP to "exchange the objects on levels 1 and 2", you aren't really exchanging the objects, you're just exchanging the pointers.

I've listed the stack commands in the following order:

```
DEPTH
ROLL
ROLLD
ROT
UNROT (added with the 49 series)
SWAP
PICK
PICK3 (added with the 49 series)
OVER
DUP
DUPDUP (added with the 49 series)
DUPN
DUP2
NDUPN (added with the 49 series)
DROPN
DROP2
DROP
; (added with the 49 series)
CLEAR
NIP (added with the 49 series)
UNPICK (added with the 49 series)
```

Two more that I wouldn't really consider to be "stack commands", but which are often used with the stack.

```
\->LIST
LIST\->
```

The 48 and 49 series also provide various operations in an "interactive stack". When the "standard stack display" is active (no "special environment" active) this can be invoked by pressing the CursorUp key, and in the 49 series, even

when in a special environment, by pressing the HIST key. Some stack operations, such as ECHO, may be available with a menu key in various special environments. I don't cover these operations in this article, but I do recommend that you experiment with them.

A command for obtaining information about the stack:

- **DEPTH**

DEPTH -> n

DEPTH doesn't manipulate the stack; it simply tells you how many levels existed before you executed DEPTH, and can be useful with some of the stack manipulation commands. Perhaps it's worth noting that executing the DEPTH command increases the stack depth by 1, but usually that new level-1 number will be used as an argument for the next command. For example, to copy the topmost stack level to level 1 (lifting the previous stack by one level), use DEPTH PICK. To move the topmost stack level down to level 1 while lifting the rest of the stack, use DEPTH ROLL. To move level 1 to the top of the stack while moving the rest of the stack down, use DEPTH ROLLD. DEPTH can also be used for testing whether the stack has anything on it; if the stack is empty, then DEPTH returns the real number 0, which the UserRPL conditional commands treat as "false"; otherwise it returns a non-zero real number, which the UserRPL conditional commands treat as "true".

DEPTH has no effect on what LAST or LASTARG returns.

Commands intended for moving stack levels:

- **ROLL**

$obj_n \dots obj_1$ n ROLL -> $obj_{n-1} \dots obj_1 obj_n$

$obj_2 obj_1$ 2 ROLL -> $obj_1 obj_2$

obj 1 ROLL -> obj

0 ROLL ->

ROLL first removes its numeric argument n from level 1 (so of course the rest of the stack moves down one level), then moves level n to level 1, while moving all levels below n upward by one level. 1 ROLL ends up leaving the stack the same as it was. 0 ROLL also ends up leaving the stack the same as it was, and doesn't error out even with an empty stack.

If you start with the stack like this:

```
4:      "D"
3:      "C"
2:      "B"
1:      "A"
```

After 3 ROLL, the stack looks like this:

```
4:      "D"
3:      "B"
2:      "A"
1:      "C"
```

n ROLL has the same effect as doing n ROLLD $n-1$ times.

The effect of n ROLL could be achieved by:

```

n
\<<
  \-> n
  \<<
    1. n 1. -
    START
    n ROLLD
    NEXT
  \>>
\>>

```

After n ROLL, LAST or LASTARG returns n .

• ROLLD

$obj_n \dots obj_2 obj_1 n$ ROLLD $\rightarrow obj_1 obj_n \dots obj_2$

$obj_2 obj_1 2$ ROLLD $\rightarrow obj_1 obj_2$

$obj 1$ ROLLD $\rightarrow obj$

0 ROLLD $\rightarrow$

ROLLD first removes its numeric argument n from level 1, then moves level 1 to level n , while moving all levels below n downward by one level. 1 ROLLD ends up leaving the stack the same as it was. 0 ROLLD also ends up leaving the stack the same as it was, and doesn't error out even with an empty stack.

If you start with the stack like this:

```

4:      "D"
3:      "C"
2:      "B"
1:      "A"

```

After 3 ROLLD, the stack looks like this:

```

4:      "D"
3:      "A"
2:      "C"
1:      "B"

```

n ROLLD has the same effect as doing n ROLL $n-1$ times.

The effect of n ROLLD could be achieved by:

```

n
\<<
  \-> n
  \<<
    1. n 1. -
    START
    n ROLL
    NEXT
  \>>
\>>

```

After n ROLLD, LAST or LASTARG returns n .

- n ROLL and n ROLLD reverse each other's effect.

• ROT

$obj_3 obj_2 obj_1$ ROT $\rightarrow obj_2 obj_1 obj_3$

ROT moves level 3 to level 1, lifting the original levels 1 and 2 one level.

ROT does the same as 3 ROLL or 3 ROLLD 3 ROLLD, or on the 49 series, UNROT UNROT.

After ROT, LAST or LASTARG returns whatever was originally on levels 3, 2, and 1.

- **UNROT** (added with the 49 series)

$obj_3\ obj_2\ obj_1\ UNROT \rightarrow obj_1\ obj_3\ obj_2$

UNROT moves level 1 to level 3, moving the original levels 2 and 3 down 1 level.

UNROT does the same as 3 ROLLD or ROT ROT or 3 ROLL 3 ROLL.

After UNROT, LAST or LASTARG returns whatever was originally on levels 3, 2, and 1.

- ROT and UNROT reverse each other's effect.

- **SWAP**

$obj_2\ obj_1\ SWAP \rightarrow obj_1\ obj_2$

SWAP exchanges levels 1 and 2.

SWAP does the same as either 2 ROLL or 2 ROLLD.

After SWAP, LAST or LASTARG returns whatever was originally on levels 2 and 1.

Commands intended for copying stack levels:

- **PICK**

$obj_n\ \dots\ obj_1\ n\ PICK \rightarrow obj_n\ \dots\ obj_1\ obj_n$

$obj_2\ obj_1\ 2\ PICK \rightarrow obj_2\ obj_1\ obj_2$

$obj\ 1\ PICK \rightarrow obj\ obj$

$0\ PICK \rightarrow$

PICK first removes its numeric argument n from level 1 (moving the rest of the stack down) and then copies level n to level 1, lifting anything that was already on the stack by one level. 0 PICK ends up leaving the stack the same as it was, and doesn't error out even with an empty stack.

If you start with the stack like this:

```
4:      "D"
3:      "C"
2:      "B"
1:      "A"
```

After 3 PICK, the stack looks like this:

```
5:      "D"
4:      "C"
3:      "B"
2:      "A"
1:      "C"
```

After n PICK, LAST or LASTARG returns n .

- **PICK3** (added with the 49 series)

obj₃ obj₂ obj₁ PICK3 -> *obj₃ obj₂ obj₁ obj₃*

PICK3 copies level 3 to level 1, lifting the previous stack by 1 level.

PICK3 has the same effect as 3 PICK.

After PICK3, LAST or LASTARG returns whatever was originally on levels 3, 2, and 1.

- **OVER**

obj₂ obj₁ OVER -> *obj₂ obj₁ obj₂*

OVER copies level 2 to level 1, lifting the previous stack by 1 level.

OVER has the same effect as 2 PICK.

After OVER, LAST or LASTARG returns whatever was originally on levels 2 and 1.

- **DUP**

obj DUP -> *obj obj*

DUP duplicates level 1, lifting the previous stack by 1 level.

DUP has the same effect as 1 PICK or 1 DUPN.

After DUP, LAST or LASTARG returns whatever was originally on level 1.

- **DUPDUP** (added with the 49 series)

obj DUPDUP -> *obj obj obj*

DUPDUP has the same effect as DUP DUP or 1 PICK 1 PICK; in other words, it makes 2 copies of level 1, lifting the previous stack by 2 levels.

After DUPDUP, LAST or LASTARG returns whatever was originally on level 1.

- **DUPN**

obj_n ... obj₁ n DUPN -> *obj_n ... obj₁ obj_n ... obj₁*

obj₂ obj₁ 2 DUPN -> *obj₂ obj₁ obj₂ obj₁*

obj 1 DUPN -> *obj obj*

0 DUPN ->

DUPN first removes its numeric argument *n* from level 1, and then it duplicates the lowest *n* levels, lifting anything above them by *n* levels. 0 DUPN ends up leaving the stack the same as it was, and doesn't error out even with an empty stack.

If you start with the stack like this:


```
4:      "D"
3:      "C"
2:      "B"
1:      "A"
```

After 2 DUPN, the stack looks like this:

```
6:      "D"
5:      "C"
4:      "B"
3:      "A"
2:      "B"
1:      "A"
```

n DUPN has the same effect as doing n PICK n times.

The same result could be achieved with:

```
n
\<<
  \-> n
  \<<
    1. n
    START
    n PICK
    NEXT
\>>
```

After n DUPN, LAST or LASTARG returns n .

• DUP2

$obj_2\ obj_1\ \text{DUP2} \rightarrow obj_2\ obj_1\ obj_2\ obj_1$

DUP2 duplicates levels 1 and 2, lifting the previous stack by 2 levels.

DUP2 has the same effect as 2 DUPN, or OVER OVER, or 2 PICK 2 PICK.

After DUP2, LAST or LASTARG returns whatever was originally on levels 2 and 1.

A command intended to push an object to the stack a number of times, and return the count:

• NDUPN (added with the 49 series)

$obj\ n\ \text{NDUPN} \rightarrow obj\ \dots\ obj\ n$

$obj\ 2\ \text{NDUPN} \rightarrow obj\ obj\ 2.$

$obj\ 1\ \text{NDUPN} \rightarrow obj\ 1.$

$obj\ 0\ \text{NDUPN} \rightarrow 0.$

NDUPN first removes both arguments from the stack, then pushes obj to the bottom of the stack n times, and then pushes the count n as a real number to level 1, lifting the rest of the stack $n+1$ levels. The result is that you end up with a total of n copies of obj in levels 2 through $n+1$, and n as a real number in level 1. $obj\ 0\ \text{NDUPN}$ just returns a count of 0., without pushing obj to the stack at all.

If you start with the stack like this:

```
4:      "D"
3:      "C"
```

```
2:      "B"
1:      "A"
```

After "New" 3 NDUPN, the stack looks like this:

```
8:      "D"
7:      "C"
6:      "B"
5:      "A"
4:      "New"
3:      "New"
2:      "New"
1:      3.
```

The same effect could be achieved with:

```
obj
n
\<<      @ Begin program.
0.        @ Initial value for 'c' (a loop counter).
\>-> o n c @ Bind local variables.
\<<      @ Begin local variables defining procedure.
0.        @ Push 0. (initial loop count) to the stack.
WHILE     @ Begin loop test.
  n <      @ Compare loop count with n.
REPEAT    @ If true, begin REPEAT clause.
  0        @ Push a copy of object to the stack.
  'c' INCR @ Increment loop count, leaving a copy of the new value on the stack.
END        @ End of loop.
c         @ Push final value of loop count to the stack.
\>>      @ Abandon local variables.
\>>      @ End program.
```

After *obj n* NDUPN, LAST or LASTARG returns both *obj* and *n*.

Commands intended for removing stack levels:

- DROPN**

obj_n ... obj₁ n DROPN ->

obj₂ obj₁ 2 DROPN ->

obj 1 DROPN ->

0 DROPN ->

DROPN first removes its numeric argument *n* from level 1, then discards levels 1 through *n*. Anything above level *n* moves down *n* levels. 0 DROPN leaves the stack the same as it was and doesn't error out even with an empty stack.

If you start with the stack like this:

```
4:      "D"
3:      "C"
2:      "B"
1:      "A"
```

After 3 DROPN, the stack looks like this:

```
1:      "D"
```

After *n* DROPN, LAST or LASTARG returns *n*.

- DROP2**

obj₂ obj₁ DROP2 ->

DROP2 discards levels 1 and 2. Anything above that was above level 2 moves down 2 levels.

DROP2 does the same as 2 DROPN.

After DROP2, LAST or LASTARG returns whatever was originally on levels 2 and 1.

- **DROP**

obj DROP ->

DROP discards level 1. Anything that was above level 1 moves down 1 level.

DROP does the same as 1 DROPN.

After DROP, LAST or LASTARG returns whatever was originally on level 1.

- **;** {added with the 49 series)

(A command represented in UserRPL by the semicolon character.)

obj ; ->

;->

; does the same as DROP if there's anything on the stack, or if the stack is empty, then it does nothing. I think that this was intended for use in ALG mode, but it works in RPN mode. The same result could be achieved by:

`\<< IF DEPTH THEN DROP END \>>`

or:

`\<< IFERR DROP THEN END \>>`

If the stack was originally non-empty, after ;, LAST or LASTARG returns whatever was originally on level 1. If the stack was originally empty, then ; has no effect on what LAST or LASTARG returns.

- **CLEAR**

obj_n ... obj₁ CLEAR ->

CLEAR ->

CLEAR simply discards all stack levels. Doesn't error out with an empty stack.

CLEAR does the same as DEPTH DROPN.

CLEAR has no effect on what LAST or LASTARG returns.

- **NIP** (added with the 49 series)

obj₂ obj₁ NIP -> obj₁

NIP leaves level 1 as is and discards level 2. Anything that was above level 2 moves down 1 level.

The effect of NIP can be achieved with SWAP DROP.

If you start with the stack like this:

```
4:      "D"
3:      "C"
2:      "B"
1:      "A"
```

After NIP, the stack looks like this:

```
3:      "D"
2:      "C"
1:      "A"
```

After NIP, LAST or LASTARG returns whatever was originally on levels 2 and 1.

A command intended for replacing an existing level:

- **UNPICK** (added with the 49 series)

obj_n ... obj₁ obj n UNPICK -> obj ... obj₁

obj₂ obj₁ obj 2 UNPICK -> obj obj₁

obj₁ obj 1 UNPICK -> obj

obj 0 UNPICK ->

UNPICK first takes both arguments *obj* and *n* from the stack and then replaces level *n* with *obj*, without moving the rest of the stack. *obj 0 UNPICK* leaves the stack as it was, and doesn't error out even with an empty stack.

For example, if you start with the stack like this:

```
4:      "D"
3:      "C"
2:      "B"
1:      "A"
```

After "New" 3 UNPICK, it will look like this:

```
4:      "D"
3:      "New"
2:      "B"
1:      "A"
```

The same result (but with slower execution) could be achieved by:

```
obj
n
\<<
  DUP          @ Copy of n.
  2. + ROLL    @ Move down the object to be replaced.
  DROP         @ Discard it.
  ROLL         @ Move the replacement up to level n.
\>>
```

After *n* UNPICK, LAST or LASTARG returns *n*.

List commands often used with the stack:

- **\->LIST**

obj_n ... obj₁ n \->LIST -> { obj_n ... obj₁ }

obj₂ obj₁ 2 \->LIST -> { *obj₂ obj₁* }

obj 1 \->LIST -> { *obj* }

0 \->LIST -> { }

\->LIST takes its numeric argument *n* from the stack, and then puts objects from level *n* down through level 1 into a list, moving the rest of the stack down by *n*-1 levels. 0 \->LIST puts an empty list on level 1, lifting the rest of the stack by 1 level.

For example, if you start with the stack like this:

```
4:          "D"
3:          "C"
2:          "B"
1:          "A"
```

After 3 \->LIST, it will look like this:

```
2:          "D"
1:  { "C" "B" "A" }
```

After *n* \->LIST, LAST or LASTARG returns *n*.

- **LIST\->**

{ *obj_n ... obj₁* } LIST\-> -> *obj_n ... obj₁* *n*

{ *obj₂ obj₁* } LIST\-> -> *obj₂ obj₁* 2.

{ *obj* } LIST\-> -> *obj* 1.

() LIST\-> -> 0.

LIST\-> puts the contents of a list on the bottom of the stack, and then pushes a count of the objects to level 1 as a real number, lifting the rest of the stack by the number of object in the list.

For example, if you start with the stack like this:

```
2:          "D"
1:  { "C" "B" "A" }
```

After LIST\->, it will look like this:

```
5:          "D"
4:          "C"
3:          "B"
2:          "A"
1:          3.
```

After LIST\->, LAST or LASTARG returns whatever was previously on level 1.

- A list "exploded" onto the stack as a series of objects followed by the count, such as the above, can be considered a "meta-object", which can be re-combined into a list by the \->LIST command.

I believe that that pretty well covers all of the UserRPL stack commands.

Of course, SysRPL has a lot more stack commands available, so sometimes with SysRPL you can achieve your goal with fewer stack manipulations, saving time and memory usage. As with most things when using RPL, a problem may be that there are so many different ways to accomplish the same thing.

This isn't really what I'd call "stack manipulation", but it seems related and can be useful. DEPTH \->LIST puts the contents of the entire stack into a list that can be stored in a variable for safe-keeping, so you can start working on something else with a "clean slate" without losing what you were working on. This can also be useful in a program when you don't have an easy way of knowing how many objects will be put on the stack; of course the DEPTH command can tell you where the top of the stack is, and thus how many objects you've put on the stack. When you want to restore the saved stack (inserted below anything on the current stack), recall the list to the stack and do LIST\->DROP. With SysRPL, you can do similar things (with the 49 series) with the "virtual stack" commands.

For "interactive use" (not within a program), assuming that "last stack" saves are enabled, an alternative way to save a copy of the current stack is to do a HALT to save it as the "last stack" within a "suspended environment", then CLEAR to get your "clean slate" for working on something else, and then when you want to restore your saved stack, do CONT immediately followed by the UNDO operation (on a 48SX/S, LeftShift STACK, above the 2 key) to replace the current stack with the saved stack. But note that if you press ENTER (or any key that invokes an implicit ENTER) between CONT and UNDO, then the saved stack will be lost.

Regards,
James

Edited: 26 Sept 2007, 12:23 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

A Third Order Complex Linear Equation Solver for the hp 33s

Posted by [Palmer O. Hanson, Jr.](#) on 20 June 2007, 10:27 p.m.

Introduction

In response to publication of my "Sixth Order Linear Equation Solver for the hp 33s" (Article No. 676) correspondent Gerson W. Barbosa wrote "... Now, a fourth or even a third order Complex Linear Equation Solver would be great, but that would be asking too much, I recognize." This article offers a third order solver.

The program which follows is not elegant. Users will see a lot of repeated code which could surely be compressed with subroutines. I didn't use many subroutines. Subroutines use labels, and labels are in short supply on the hp 33s. Program memory is not in short supply on the hp 33s.

Program Description

The program is essentially a modification of my "A Label Saving Simultaneous Equation Solution for the 33s" (Article No. 599) to use the complex arithmetic capability of the hp 33s. The solution is obtained using Cramer's rule. The L routine sets up initial conditions. The M routine provides input, sequencing through the subroutines and output. The D subroutine calculates the desired determinant. The V and X subroutines provide a circular shifting of the stored input data to set up the matrix needed for a particular determinant calculation.

The input and display of complex numbers of the form $x + iy$ follows the convention established in section 9 of the manual. The imaginary part is in y (the upper level of the display) and the real part is in x (the lower level of the display).

The input part of the program mimics the INPUT function for single variables. A prompt which indicates the row and column to be entered is flashed; for example, the prompt "1,01" asks for the coefficient from the first row and first column and the prompt "2,04" asks for the second element of the vector. After the prompt the program stops with the previously entered complex number in the display. If the values displayed are the desired ones the user simply presses R/S to accept those values and continues on. Thus, the user can rerun the previously entered problem, say, because of a suspicion that some data may have been entered incorrectly.

User Instructions

1. Press XEQ L to start the program.
2. The display will flash "1.01" to request the complex coefficient from the first row and first column of the augmented matrix and stop with previously entered data in the display. Press R/S to accept the displayed data. To enter a new complex coefficient enter the imaginary part, press ENTER, enter the real part and press R/S.
3. The display will flash "1.02" to request the complex coefficient from the first row and second column of the augmented matrix, and so on. When the display flashes "1.04" enter the complex coefficient from the first row and fourth column of the augmented matrix, which is the first element of the vector.

4. Continue to enter the remaining coefficients in response to the prompts. If you fail to see a prompt you can recover it by pressing Roll Down.
5. When you press R/S after entering the data required in response to the prompt "3.04" the program will proceed with the solution. If the determinant of the input matrix is zero program execution will stop with the message "DIVIDE BY ZERO" in the display.
6. If the determinant of the input matrix is not zero the program will stop with the determinant of the matrix in the display as $x + iy$ and also in variables Y and Z. The augmented matrix will be in its original position in variables A through X. The solution vector will be in the six statistics registers.
7. Press R/S to see the first element of the solution vector.
8. Press R/S to see the second element of the solution vector.
9. Press R/S to see the third element of the solution vector.
10. The solution values can be recalled for further use with RCL (i) commands with i from 28 through 33 or more simply through use of the options of the SUMS menu as illustrated in program steps M088 through M0096

Program Listing

L0001	LBL L	
L0002	CF 4	
L0003	1.024	
L0004	STO i	
L0005	CL Sum	
L0006	1.01	
L0007	Sum+	
M0001	LBL M	
M0002	FIX 2	
M0003	RADIX,	
M0004	Sumx	From the SUMS menu
M0005	ENTER	
M0006	PSE	
M0007	FIX 9	
M0008	RADIX.	
M0009	RCL (i)	
M0010	1	
M0011	STO+ i	
M0012	RD	Roll Down
M0013	RCL (i)	
M0014	STOP	
M0015	STO (i)	
M0016	1	
M0017	STO- i	
M0018	RD	
M0019	x<>y	
M0020	FS? 4	
M0021	0	
M0022	STO (i)	
M0023	2	
M0024	STO+ i	
M0025	0.01	
M0026	Sum+	
M0027	0.05	
M0028	Sumx	From the SUMS menu
M0029	FP	
M0030	x ne y?	Is x not equal to y ?
M0031	GTO M	
M0032	0.96	
M0033	Sum+	
M0034	1	
M0035	STO- i	
M0036	ISG i	
M0037	GTO M	
M0038	XEQ D	
M0039	29	
M0040	STO i	
M0041	RD	
M0042	STO (i)	

M0043 DSE i
M0044 x<>y
M0045 ST0 (i)
M0046 CMPLX 1/x
M0047 XEQ V
M0048 XEQ D
M0049 33
M0050 ST0 i
M0051 RD
M0052 n From the SUMS menu
M0053 Sumx From the SUMS menu
M0054 CMPLX /
M0055 ST0 (i)
M0056 DSE i
M0057 x<>y
M0058 ST0 (i)
M0059 XEQ V
M0060 XEQ D
M0061 31
M0062 ST0 i
M0063 RD
M0064 n From the SUMS menu
M0065 Sumx From the SUMS menu
M0066 CMPLX /
M0067 +/-
M0068 ST0 (i)
M0069 DSE i
M0070 x<>y
M0071 +/-
M0072 ST0 (i)
M0073 XEQ V
M0074 XEQ D
M0075 29
M0076 ST0 i
M0077 RD
M0078 n From the SUMS menu
M0079 Sumx From the SUMS menu
M0080 CMPLX /
M0081 ST0 (i)
M0082 DSE i
M0083 x<>y
M0084 ST0 (i)
M0085 XEQ V
M0086 XEQ D
M0087 STOP
M0088 n From the SUMS menu
M0089 Sumx From the SUMS menu
M0090 STOP
M0091 Sumy From the SUMS menu
M0092 Sum x^2 From the SUMS menu
M0093 STOP
M0094 Sum y^2 From the SUMS menu
M0095 Sum xy From the SUMS menu
M0096 STOP

D0001 LBL D
D0002 RCL A
D0003 RCL B
D0004 RCL K
D0005 RCL L
D0006 CMPLXx
D0007 RCL U
D0008 RCL V
D0009 CMPLXx
D0010 ST0 Z
D0011 x<>y
D0012 ST0 Y
D0013 RCL C
D0014 RCL D
D0015 RCL M
D0016 RCL N
D0017 CMPLXx
D0018 RCL Q
D0019 RCL R
D0020 CMPLXx
D0021 ST0+ Z
D0022 x<>y
D0023 ST0+ Y
D0024 RCL E

D0025 RCL F
D0026 RCL S
D0027 RCL T
D0028 CMPLXx
D0029 RCL I
D0030 RCL J
D0031 CMPLXx
D0032 ST0+ Z
D0033 x<>y
D0034 ST0+ Y
D0035 RCL E
D0036 RCL F
D0037 RCL K
D0038 RCL L
D0039 CMPLXx
D0040 RCL Q
D0041 RCL R
D0042 CMPLXx
D0043 ST0- Z
D0044 x<>y
D0045 ST0- Y
D0046 RCL A
D0047 RCL B
D0048 RCL S
D0049 RCL T
D0050 CMPLXx
D0051 RCL M
D0052 RCL N
D0053 CMPLXx
D0054 ST0- Z
D0055 x<>y
D0056 ST0- Y
D0057 RCL C
D0058 RCL D
D0059 RCL I
D0060 RCL J
D0061 CMPLXx
D0062 RCL U
D0063 RCL V
D0064 CMPLXx
D0065 ST0- Z
D0066 x<>y
D0067 ST0- Y
D0068 RCL Y
D0069 RCL Z
D0070 RTN

V0001 LBL V
V0002 1.03002
V0003 ST0 i
V0004 RCL G
V0005 XEQ X
V0006 5
V0007 ST0- i
V0008 RCL H
V0009 XEQ X
V0010 1
V0011 ST0+ i
V0012 RCL O
V0013 XEQ X
V0014 5
V0015 ST0- i
V0016 RCL P
V0017 XEQ X
V0018 1
V0019 ST0+ i
V0020 RCL W
V0021 XEQ X
V0022 5
V0023 ST0- i
V0024 RCL X
V0025 XEQ X
V0026 RTN

X0001 LBL X
X0002 x<> (i)
X0003 ISG 1
X0004 x<> (i)

```
X0005   ISG  i
X0006   x<> (i)
X0007   ISG  i
X0008   STO (i)
X0009   RTN
```

Test problem:

The first number in each coefficient is the real part. The second number in each part is the imaginary part. But, remember that the imaginary part is entered first in the program.

+1, +3	-1, +2	+4, 0	+10, -8
-3, -2	-5, +1	+1, +1	+10, -4
+2, 0	0, +3	-3, +4	+25, +12

The determinant of the matrix is +155,-66 and the solution vector is

+3, +1
-2, -2
+1, -4

To solve a linear equation with real coefficients:

The primary purpose for this program is the solution of third order linear equations with complex coefficients. The program can be used to solve third order linear equations with real coefficients by simply entering a zero for each imaginary part during input. The FS? 4 0 sequence at M0020/M0021 provides an easier method of operation when using real coefficients. When the program stops to receive the first input after the prompt "1.01" set flag 4 before proceeding with input. Then it is only necessary to enter the real value at each input point since with flag 4 set the program will provide the zero for the imaginary part of each input. Thus, if this program is on your machine there is no need to have a program such as that in Article 599. You get both capabilities at a cost of one additional label. Note that flag 4 is reset at the beginning of the program.

Edited: 26 June 2007, 9:47 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

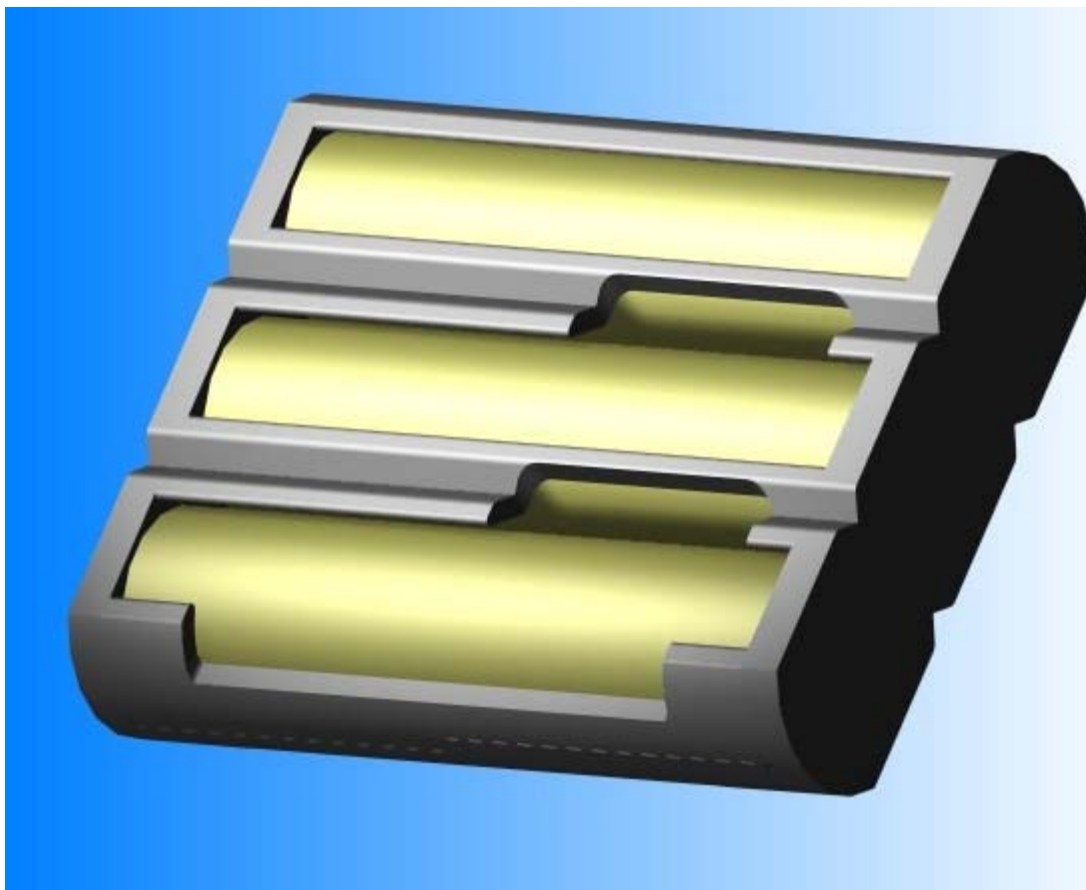
[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Making replacements for 82001 Classic Battery

Posted by [Don Shaffer](#) on 20 Apr 2007, 6:01 p.m.

Building Replacement Classic Battery Packs



The basic HP Classic type battery 82001 is shown as a 3D model. Going from just having some of the old batteries to actually being able to make replacements was an interesting project for me. At first I studied the existing design spending much time making sketches how I could fabricate the covers. I had four old batteries available. One I had purchased as a spare for some tests where I needed to make sure I had a backup. Another was the original battery pack I had from my old HP-75. I also had two packs removed from an older HP voltmeter that used both of them. I had used up the life on the hP-75 battery once and had to cut the battery cover open to replace the cells. This is a typical battery repair that has been done and steps for doing this given in the museum archives. The main issue with this was that you had to purchase a perfectly good AT&T cell phone type battery and cut it apart; then solder the old contact strips from the original battery carefully onto the new cell pack. Then the whole thing is hot glued back together again. You will find that cutting the covers apart into the two halves is not as easy as it seems, as you can ruin your old covers if you make the wrong movement, or alternatively cut yourself. Also, after reading about the Nickel Cadmium cells you find that it is possible to damage them if they get too hot while

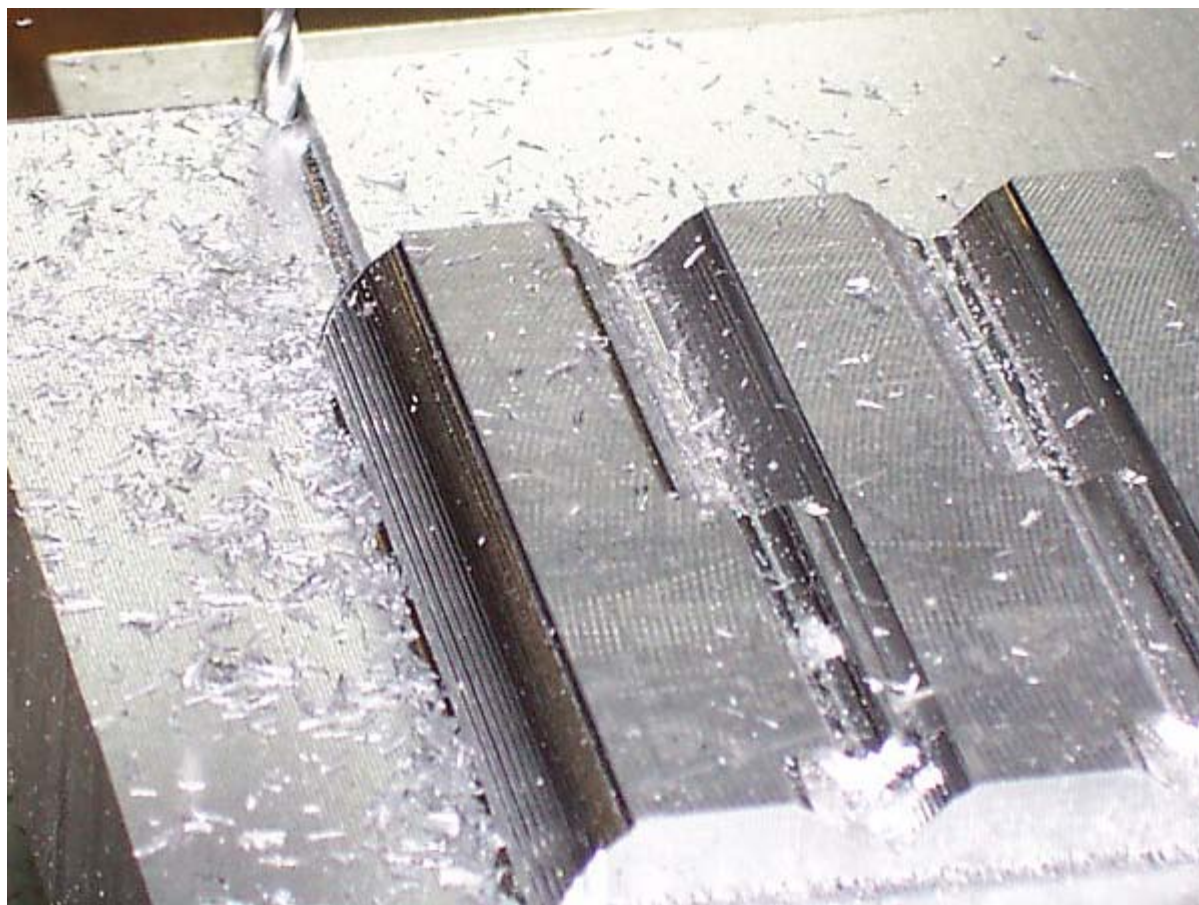
soldering the contacts on. Again while doing this you should wear safety glasses in case you cause the cells to overpressurize. I wanted to find a better solution for this problem also. After searching the web I found that the normal way the cell straps and contact straps are connected to the cells is by using a battery tab welder. I understand that this service may be a lot more common than I had thought, so you may have a some local service shop that does this sort of thing right in your own city.

Going back to the other problem of molding the covers; at first, I tried some really basic methods taken from art and craft books. I tried making a reverse mold several times without any success. These attempts included plaster cast molding over the old cover half, using heat cured molding clay, poured rubber molding, and finally liquid metal mold material was tried. This last experiment turned out as a disaster, since the solvent in the liquid metal actually dissolved one of my original plastic battery halves.

What to do next? Well I thought about asking for a mold making service, but when I looked into this, costs were pretty steep beyond what I could afford. So the next step was to try to find out how I could learn to make these myself using the plastic injection molding techniques.

I have had some CAD experience so modeling was not totally new to me. However getting the right software was the first item. One quickly finds that the costs of these 3D modeling packages can also be quite expensive. And after one prepares a mold design, then how do you go about getting it into the piece of metal? I first tried making the 3D models from basic sketches of the battery I wanted to make.

Making A Replacement Battery Mold

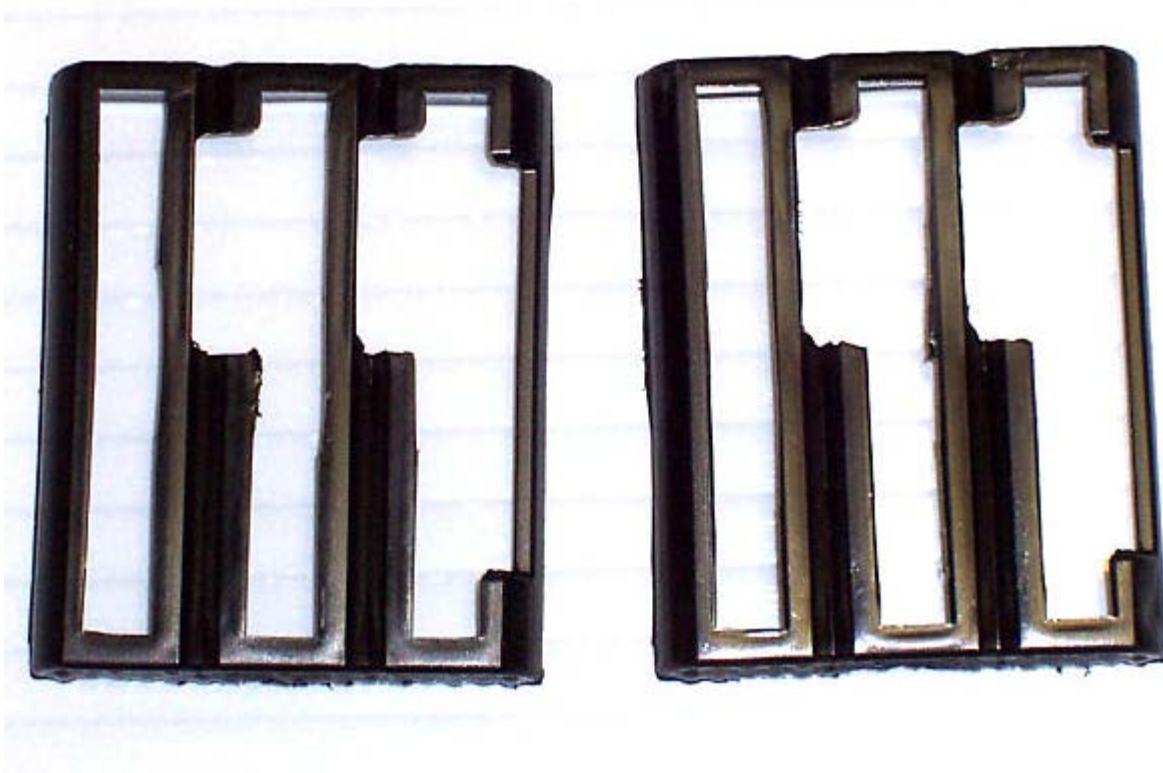


Getting to the point where I actually am cutting the mold halves took several years. This included learning more about how to do 3D solid design of molds, how to translate these solid models into CNC code, and then how to run the CNC milling machine to actually produce the mold halves. Along the way one has to obtain several important items, CNC tool path geometry software to translate the solid model to CNC 'G-code'. The actual CNC milling machine needs to be obtained or put together from parts and bits purchased from ebay the way I did it. Then tooling is also needed. You also need to have the CNC controller and oh yes, a computer to run the controller that takes the G-codes and sends them to the machine during the milling operations. As you can see in the photo I finally reached this point.

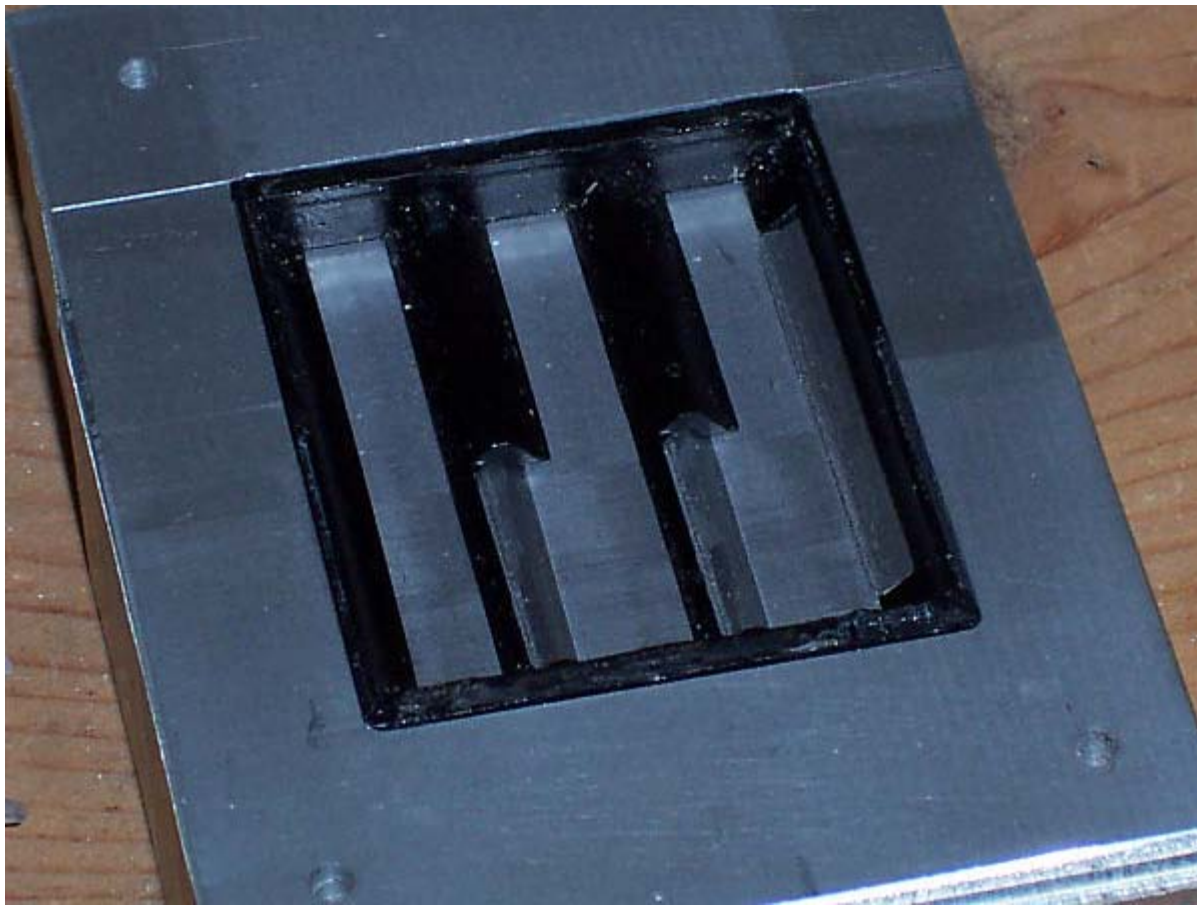
The next item on the list was getting a plastic press to inject the molds once they were finished. Again, after some

research a book of plans was found and the press was built from scratch using angle iron, and other bar stock. Finally the launch day for pressing the first prototype part finally arrived.

Pressing First Battery Parts

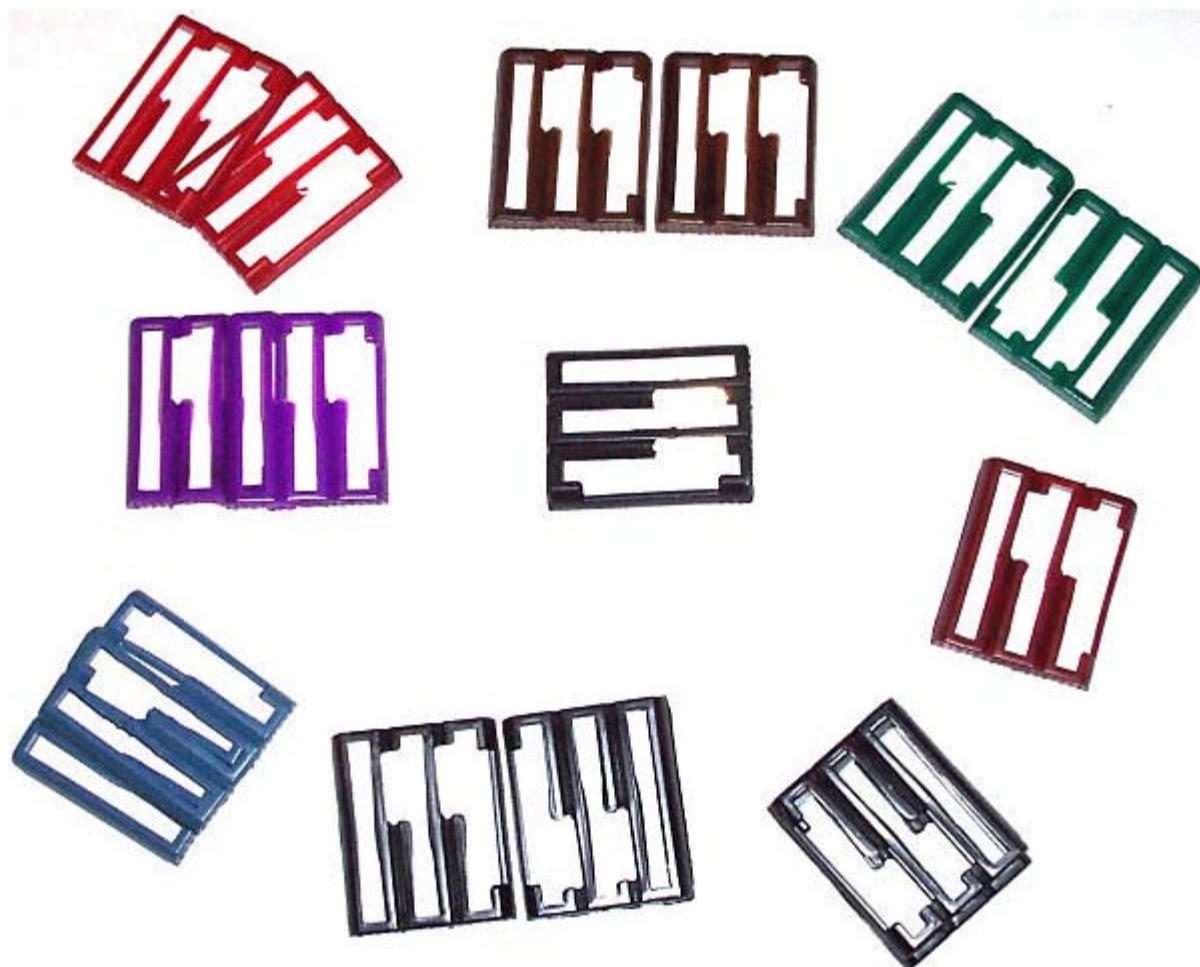


Your basic black molded covers are shown here. I also had made a fixture to check the size of the battery cover half.



This allows me to check that the covers fit closely to the original shape of a Classic stype 82001 I wanted to replace. Of course the design is different in several ways. The original design has an intricate overlapping interlocking cover arrangement that is much too complicated in my opinion. Instead for the design I choose much simpler flat halves easily hot glued, or later cut open for cell replacement. Original packs also have fancy mortice and tenon style joints at each end apparently interlocking the halves tightly and making it difficult to cut apart. These must have been hot air welded together or other means as the end joints are barely discernable. I had also run the CNC machine mold cutting as fast as possible while still getting proper geometry. The tool had some slight overshoot of the cutter making mold for the outer surface. At first I had planned to redo this with a new mold with this feature eliminated, but after pressing the first set of prototype parts I decided I liked the ribbed effect on the ends. This also differentiates my design from the original, so collectors can tell the new replacement parts from any of the original stock Classic packs marked with the part number and mold tool numbers.

Colors Too!



Next was some experimentation using whatever items of plastic I could find for testing. Also getting some different colors as you see. So in addition to the standard black I could get red, green, blue, chocolate, or other colors. While I do not know how popular these would be, I tried what I could for pilot production runs.

Making the Contacts



Next and final part of the project for me, was figuring out how to make the battery terminal contacts. I had some of the originals to figure out the right size. After some investigation on the web you find out that the basic material used for this type part is solid Nickel sheet metal. The use of copper contacts is not the best choice since copper can corrode much easier, while the nickel is much more resistant to corrosion. The nickel comes as sheet, this of course must first be cut into strips, then shaped and formed to fit into the valley's formed between two AA cells on the battery pack. The longer end connection goes to one battery terminal, and a second shorter terminal strap goes connects on the other end. These of course must be put onto the correct side of the pack so the exposed terminals connect up with the proper connections in the device battery compartment. Again the details for these proper connections are given in the museum archives. Final assembly is too hot-glue the halves together once the nickel contacts and connector straps have been welded onto the cells

Contact information

I want to thank the museum for a place where I could discuss this project. Batteries and cover sets are available. I am still working out the selecting capacities that people would like to obtain. So any suggestions will be studied carefully. Also any other ideas for another project such as this, let me know too. I already had a question about making replacement battery cover doors for some of the other HP's models. Please contact me at email d_dbatteryworks@earthlink.net or if that does not work at dsmal12@earthlink.net for more information or give comments.

Edited: 21 Apr 2007, 2:41 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

programming using the HP 17bii+ solver

Posted by [Don Shepherd](#) on 14 Apr 2007, 2:48 p.m.

The HP 17bii+ calculator has several characteristics that might make it interesting to members of this community:

- it fits in your shirt pocket nicely
- it has a 2-line display
- it has 6 soft function keys
- it has user-definable menus
- it has a very uncluttered keyboard
- it has useful finance and business functions
- it can print using a small infrared printer
- you can use either RPN or algebraic mode
- it has 12-digit accuracy
- it has 30,740 bytes of user memory
- it is programmable
- it is still being manufactured!

I became interested in this model after reading somewhere that it was programmable. I determined that it is not “keystroke” programmable like its financial brother, the venerable 12c and its successors. But it is programmable via the built-in equation solver application.

I imagine that the original inventors of the equation solver never intended that it would be used as a general purpose “programming” language, just as the inventors of the 12c cash flow functions probably never imagined that anyone would use those financial functions for indirect addressing. But the great thing about HP calculators is that both those things are possible. And members of this community have always been willing to share insights and experiences.

So I bought the 17bii+ just to see what it could do, from a programming perspective. I knew that every programming language must support three structures: sequence, decision, and iteration (loops). So I began to look at the solver. Yep, it can do sequences (especially when you put “0X” next to things you don’t want to be included in the solution). Yep, it can do decisions (If (condition : do if true : do if false)) And yep, it can do loops (sigma function). And it can assign values to named variables (load function) and use those variables (get function) in calculations. The solver cannot use the 10 registers, but since you can create your own variable names, that is not a shortcoming. The solver cannot execute subroutines, GOTO a label or line number, or exit a loop early, and this makes it unacceptable as a general purpose programming language. But I found that you can still do some useful things with it.

For instance, you can sum the digits of a number:

```
sod = sigma(i:0:log(n):1:mod(ip(n):10)+0xL(n:n/10))
```

You can determine if a number is prime or not (a result of 0 means the number is prime, any other number is the first factor of the input number):

```
fact = 0xL(j:0)+
```

```
if(mod(n:2)=0:2:sigma(i:3:sqrt(n):2:
if(mod(n:i)=0:if(g(j)=0:0xL(j:1)+i:0):0)))
```

You can calculate a standard mod-10 checkdigit (maximum 12 input digits):

```
ckdig = 0xL(m:2)xL(c:-1)
+mod(10-
mod(sigma(i:0:log(n):1:
0xL(a:mod(ip(n):10)xg(m))
+if(g(a)<10:g(a):g(a)-9)
+0xL(m:g(m)+g(c))
xL(c:-g(c))
xL(n:n/10)):10):10)
```

You can convert from decimal to binary:

```
dec2bin = sigma(i:0:11:1:mod(n:2)x(10^i)+0xL(n:ip(n/2)))
```

Or from binary to decimal:

```
bin2dec = sigma(i:0:log(n):1:mod(ip(n):10)x2^i+0xL(n:n/10))
```

Or from decimal to octal:

```
dec2oct = sigma(i:0:11:1:mod(n:8)x(10^i)+0xL(n:ip(n/8)))
```

I tried to carefully type the equations above, but I'm not perfect. If you try one of these on your 17bii+ and can't get it to work, post a message on the forum and I'll help.

Now, I admit that I wouldn't attempt to do a payroll program on the 17bii+, and the lack of scientific functions would make this calculator inappropriate for many members of this community. Nevertheless, I think it is great that an application designed to solve equations can, in this case, be used for some general purpose programming tasks.

HP calculators are great!

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

HP Calculator Software- Literature Survey

Posted by [Charles](#) on 31 Mar 2007, 4:38 p.m.

Hewlett Packard Calculator Programmes- Literature Survey

This is based on the article referenced below with updates. The format is not ideal- it was originally a spreadsheet.

Reference "Published Calculator Programs for Chemical Engineers" John.R.Garrett Chemical Engineering March 7 1983 pg 149 to 160

Abbreviations

CE = Chemical Engineering McGraw-Hill

HP = Hydrocarbon Processing

OGJ = Oil and Gas Journal Pennwell

PEI= Petroleum Engineer International

The information is sorted as follows: Title, Author, Type of Calculator, Journal ,Volume ,Date ,Pages ,Notes

Solving Engineering Problems on Programmable Pocket Calculators: Stripping and Flashing: 1: Binary Distillation 2: Multicomponent Flash Benenati, R.F 67/97 CE 84 14-Mar-77 129-132

Streamline Flash Computations with a Calculator Program Mansouri, S. 67/97 CE 86 27-Aug-79 99-101

Flash Calculations for Plant Operations Eased Parker, R.F. and Oliver, I.H. 67/97 OGJ 77 17-Dec-79 76-79

Hand Calculator Program Speeds Flash Calculations Oliver, I.H. 67/97/41 OGJ 78 31-May-80 130-132

Speed of Hand-calculator programs can be Improved Kostecke, S.T. 67/97 OGJ 78 11-Aug-80 107-110

Rapid Calculator Solutions ASTM/TBP other Probability Plot Problems Denchfield, T.D. 67/97 OGJ 79 27-Apr-81 179-184

Crude Dehydration / Desalting Calculations Tran, V.B. 41 OGJ 80 15-Mar-82 76-79

Calculator Program Finds Petroleum Fraction Viscosities Over Wide Temperature Range Barnea, G.M. 67/97 OGJ 80 10-May-82 148-150

Programmable Calculator Speeds Pipeline Span Computations Marks, A. 67/97 OGJ 76 09-Jan-78 106-107

Hand held Calculator Programs for frequently used Formulas: Part 1 Williams-Hazen Pressure drop Turner, W.J. 67/97

PEI 51 01-May-79 84-90

Programmable Calculator Speeds Gas Line Calculations Parker, R.F. 67/97 OGI 77 07-May-79 67-72

Estimating Products Line Commingling Marks, A. 67/97 OGI 77 19/11/79 109-110

Pipeline Liquid flow Problems solved by Calculator Burnett, R.R. 67/97 OGI 77 19/11/79 134-152

Calculators can ease pipeline surge analysis - part1 Hein, M 41 OGI 79 10-Aug-81 100-106

Analysing Line Surge problems with Hand-held calculator- Part 2 Hein, M 41 OGI 79 17-Aug-81 128-134

Nonlinear Regression on a pocket calculator Clare, B.W. 41 CE 89 23-Aug-82 83-89

Polynomial regression on a pocket calculator Clare, B.W. 41 CE 89 04-Oct-82 121-124

A note on rapid determinations of present value in the presence of growth and inflation Papendorf, O.L 41 Journal of the South African Institute of Mining and Metallurgy 01-Dec-81 345-347

Project Economic Analysis Martin, K.L. and Barna, B.A. 41 CE 91 23-Jul-84 73-78

Spreadsheet program for handheld has variety of uses Coffey, L.F. 41 OGI 82 26-Mar-84 133-134

Net Cash flow profile is calculated, analyzed Coffey, L.F. 41 OGI 82 02-Apr-84 141-142

Spreadsheet Cash-flow analysis combined Coffey, L.F. 41 OGI 82 09-Apr-84 99-102

Calculator Program for Compressible flow in pipes Gyori, I. 67/97 CE 92 28-Oct-85 55-60

Liquids pipeline calculator prompts user for data entry Scott, L. 41 OGI 82 18-Jun-84 138-145

Program predicts pressure drop for gas flow across an orifice meter Stephens, F.A. 67/97 CE 89 29-Nov-82

Program calculates orifice size for gas flows Mink, W.H. 67/97 CE 87 25-Aug-80

Program sizes control valves for gas and vapor Monsen, J.F. 67/97 CE 90 31-Oct-83

Program sizes control valves for liquids Monsen, J.F. 67/97 CE 88 18-May-81 159-163

Calculator Program Computes Centrifugal Pump Efficiency Marks, A. 67/97 OGI 78 22-Dec-80

Sizing force mains for economy Dancs, L. 67/97 Water Sewage Works 124 01-Mar-77 84-86

Hydraulic Calculations for Small Programmable calculators Croley, T.E. 67/97 Water Sewage Works 124 01-Mar-77 64-71

Predicting sulphide in force mains Kienow, K.E. and Kienow, K.K. 67/97 Water Sewage Works 125 01-Dec-78 48-49

Relative Humidity from Psychrometric Data Stenius, A.S. 67/97 Tappi 62 01-Apr-79 87-88

Industrial Wastewater Treatment Plant Model Chacey, K.A., McAvoy, W.S. 41 Pollut. Eng 14 01-Jun-82 25-28

Solve Engineering problems on Programmable Pocket Calculators Benenati, R.F. 67/97 CE 84 28-Feb-77 201-206

Versatile Calculator Program Eases Piping design Simpson, L.L. 67/97 CE 86 29-Jan-79 105-109 1

Steam Flow in Steel pipes Bryan, T.S. and McLaury, N.T. 67/97 Tappi 62 01-Jun-79 91-92

- Computer Program Slashes Piping Analyses Time Hassouneh, M and Bhaumik, H. 67/97 OGI 77 29-Oct-79 167-172
- Pressure Loss through Valves Bagadia, K. 67/97 Plant Eng. 33 31-Oct-79 81
- Friction Head Loss in Pipe Bagadia, K. 67/97 Plant Eng. 33 31-Oct-79 82
- Analyse Fire Water Network by Calculator Bhaumik, H. 67/97 OGI 77 31-Dec-79 182-189
- Versatile Programme for pressure-drop calculations Meyer, J.H. 67/97 CE 87 10/3/191980 139-142
- Calculation of Drop Leg Performance Dougherty, S.J. 67/97 Tappi 63 01-Dec-80 115-116
- Finding Volume in Partially Filled Tanks Santi, E. 67/97 CE 86 18-Jun-79 144-147
- Performing Cost-effective Analysis for Alternative Interceptor Sewer Designs Kienow, K.E. and Kienow, K.K. 67/97 Water Sewage Works 125 01-Oct-78 43-48
- Hand -Held Calculator Programs for Frequently Used Formulas: Discounted Cash Flow Projection with Price and Cost Escalation Turner, W.J. 67/97 PEI 52 01-Apr-80 76-94
- Program Calculates Stock Option Tax Oxner, E. 67/97 EDN 6 04-Feb-81 87
- Automatic Stability Calculations for Feedback Control Systems Gokbudak, M.T. 41 Control. Eng. 29 01-Jun-82 80-82
- Calculator program for Sour Water Stripper design Wild, N.H. 67/97 CE 86 12-Feb-79 103-113
- Quick Calculation of Heat Exchanger Weight Taylor, M. 67/97 Process Eng. 61 01-Jan-80
- Kinetics of Fixed Bed Sorption Processes Tan, H.K.S. 67/97 CE 87 24-Mar-80 117-119
- Calculator program aids design of spouted beds Mele, D. and Martinez, J. 67/97 CE 87 20-Oct-80 137-139
- Coating dryer calculations Walker, R.C. 67/97 Tappi 64 01-Feb-81 119-121
- Calculator program for a steam condenser Haydu, L. 67/97 CE 87 09-Feb-81 99-102
- Using a Programmable Calculator to Solve Fan Law Problems Loeffler, J.J. 67/97 Plant Eng. 35 19-Mar-81 167-170
- A New Way to Rate an Existing Heat Exchanger Herkenhoff, R.G. 67/97/41 CE 88 23-Mar-81 213-215
- Calculating the corrected LMTD in Shell and tube heat exchangers Blackwell, W.W. 67/97 CE 88 24-Aug-81 101-106
- Program Calculates Hydrocyclone Efficiency Merrill, F. 41 CE 88 02-Nov-81 71-78 2
- Program Calculates Flame Temperature Vancini, C.A. 67/97 CE 89 22-Mar-82 133-136
- Solving problems of varying heat transfer areas in Batch Processes Marzi, M. 41 CE 89 17-May-82 101-107
- Calculate enthalpy with a pocket calculator Schneider, R.T. 67/97 CE 84 23-May-77 142-152
- Black Box for Pulp Streams McLaury, N.T. 67/97 Tappi 61 01-Sep-78 117-118
- Multi- or Single Sheet Light Scattering and Absorption Coefficients Stenius, A.S. 67/97 Tappi 62 01-Jan-79 89-91 3
- Calculator gives Compressibility Factors Burnett, R.R. 67/97 OGI 77 11-Jun-79 70-74

Program Performs Vapor-Liquid Equilibrium Calculations Rice, V.L. 41 CE 89 28-Jun-82 77-86

Tristimulus Values and Chromaticity Coordinates from Reflectance Values Stenius, A.S. 67/97 Tappi 62 01-Nov-79 123-124

Evaluation of Cation Hydrolysis Schemes with a pocket calculator Clare, B.W. 67/97 J. Chem Educ. 56 01-Dec-79 784-787

Newsprint Properties- Opacity percent saturation, Dominant Wavelength and Scattering Coefficient Davidson, J.W. 67/97 Tappi 62 01-Dec-79 115-116 4

Calculated Freedom- How you can dispense with your mainframe computer Clare, B.W. 67/97 Chem. Br 16 01-May-80 249-256

Calculating Properties of Chemical Compounds Santi, E. 67/97 CE 87 02-Jun-80 75-77

Brookfield Viscosity Walker, R.C. 67/97 Tappi 63 01-Dec-80 117-119

Pinpoint Carbon Deposition Colton, J.W. 67/97 HP 60 01-Jan-81 177-184

Programs solve Underwood's Equation Ladanois, J.M. 67/97/41 HP 60 01-Apr-81 231-233

Program aids Cryogenic Solubility Calculations Lebrun, P. 67/97 CE 88 13-Jul-81 127-131 5

Water based coatings:characteristics Walker, R.C. 67/97 Tappi 64 01-Sep-81 181-185

Compute Volume Correction Factors on your Calculator Mullins, T.E. 41 HP 60 01-Nov-81 295-298

Computer predicts pipe corrosion Kienow, K.E. and Kienow, K.K. 67/97 Water Sewage Works 125 01-Sep-78 78-98

The Programmable Electronic Calculator in Underground Corrosion related activity:(High Pressure pipelines) Seifert, R.L 67/97 Mater. Perf. 19 01-Jun-80 34-38

The Programmable Electronic Calculator in Underground Corrosion related activity:(determination of an operating pressure for a corroded high pressure pipeline) Seifert, R.L 67/97 Mater. Perf. 19 01-Jul-80 26-33

The Programmable Electronic Calculator in Underground Corrosion related activity: (determines the maximum corrosion dimensions which allow the use of the assigned MAOP in a pipeline) Seifert, R.L 67/97 Mater. Perf. 19 01-Aug-80 21-30

The Programmable Electronic Calculator in Underground Corrosion related activity:(cathodic protection rectifier interference calculations) Seifert, R.L 67/97 Mater. Perf. 19 01-Sep-80 33-34

The Programmable Electronic Calculator in Underground Corrosion related activity:(calculations and storage of interference network constants) Seifert, R.L 67/97 Mater. Perf. 19 01-Oct-80 29-32

The Programmable Electronic Calculator in Underground Corrosion related activity:(comments on programs and on HP34C programs) Seifert, R.L and Wright, M.M. 67/97/34C Mater. Perf. 20 01-Jan-81 50-52

Programs to Correlate Data- Multifit Part 1 Volk, W. 67/97 CE 86 23-Apr-79 128-132

Program to fit a family of lines - Multislope Part 2 Volk, W. 67/97 CE 86 04-Jun-79 133-138

Correlating one dependent variable with two independent variables - Multi-2 Part 3 Volk, W. 67/97 CE 86 10-Sep-79 131-135

Second degree polynomial calculator program - part 4 Volk, W. 67/97 CE 86 19-Nov-79 149-153

Calculator solution to a three variable problem- part 5 Volk, W. 67/97 CE 86 17-Dec-79 93-98

Correlating the Hyperbolic Function- Part 6 Brosens, J.R. 67/97 CE 87 07-Apr-80 97-99

Calculator Program for Normal and Log-Normal Distributions Chapin, J.F. 67/97 CE 87 15-Dec-80 75-78

HP41C generates a Pseudorandom Sequence Patterson, I 41 Electronics 54 19-May-81 17

Calculate Statistics from a Histogram Thomas, W.B. 67/97 CE 88 27-Jul-81 95-96

Average and Related Data Walker, R.C. 67/97 Tappi 64 01-Nov-81 119-121

Confidence Intervals for the Odds Ratio in Case-control Studies- A Calculator program for Cornfield's method Guess, H.A. 67/97 Comput. Programs Biomed 14 01-Feb-82 73-76

HP-67 Calculates Maximum Nonlinearity Error Park, K. 67/97 Electronics 55 05-May-82 156-159

Calculator Least Squares Evaluation for k for the First Order Rate Equation DeTar, D.F. 67/97 Comput. Chem 2 No 1 1978 39-41

Calculator Least Squares Evaluation for k for the Second Order Rate Equation DeTar, D.F. 67/97 Comput. Chem 2 No 1 1978 43-45

Calculator Least Squares Evaluation of k, Y_0 and Y_{∞} for the First Order Rate Equation DeTar, D.F. 67/97 Comput. Chem 2 No 2 1978 99-104

Least Squares Treatment of the Arrhenius Equation by a Programmable Calculator DeTar, D.F. 67/97 Comput. Chem 2 No 3-4 1978 143-147

Calculator Least Squares Evaluation for k for the First Order Rate Equation DeTar, D.F. 67/97 Comput. Chem 2 No 3-4 1978 149-151

Calculator Routine for 3x3 Symmetrical Matrix DeTar, D.F. 67/97 Comput. Chem 2 No 1 1978 37-38

The Power of the HP-67 Calculator- Part 2 9 Equations, 9 Unknowns and check program Arp, R.C. 67/97 BYTE 4 01-Apr-79 176-188 6

Integer Base Conversion on Handheld Programmable Calculators Lewart, C.R. and Mockler, J. 67/97 Comput. Des. 19 01-May-80 202-207

Calculator plots time response of Inverse Laplace Transform Librach, H. 41 Electronics 54 14-Jul-81 149

Matrix diagonalization with a programmable calculator Bauer, H. 41 Comput. Chem 6 No 1 1982 15-20

MECONV Metric-English Conversions Rayner, F.A. 41 Ground Water 20 No 3/4 1982 214-216

A Calculator programme for determination of Phosphate buffer composition Mauk, G.A. and Scott, R.A. 67/97 Comput. Chem 5 No 1 1981 67-69

Simple Computer Program for a low cost Desk-top Calculator applied to the evaluation of Gas-Liquid Chromatographic Analysis of 17-Ketosteroids and Pregnanes Schwarz, S. and Stecher, W. 67/97 J. Chromatog. 223 No 2 1981 253-265

Calculator Can Aid in Piping Analysis Wheeler, W. 25 Water Sewage Works 124R 01-Jan-77 28-30

Fast programming on small calculators (Hazen- Williams formula) Wheeler, W. 25 Civ. Eng. N.Y. 47 01-Apr-77 58-60

Solve equivalent pipe problems with a small calculator Wheeler, W. 25 Water Sewage Works 124R 30-Apr-77 124-126

Hardy Cross Distribution Analysis Wheeler, W. 25 Water Sewage Works 124 30-Apr-77 130-133

Open Channel Hydraulic Calculation by Programmable Calculator: Circular Pipe, Rectangular & Trapezoidal Channel, Manning Equation Dancs, L. 25 Water Sewage Works 124 01-Dec-77 75-77

Open Channel Hydraulic Calculation by Programmable Calculator: Manning Formula for Rectangular and Trapezoidal channels, Scobey- Kutt formula Dancs, L. 25 Water Sewage Works 125 01-Jan-78 84

Flow profiles in Trapezoidal Channels Marinos, A.T. 25 ASCE J. Hydraul. Div 105 01-Jan-79 96-101

The Manning Equation Revisited Nitto, R.J. 25 Water Waste Eng. 16 01-Jun-79 30

Pocket Calculator Program for Traverse Point Location in Stack Sampling Wesolowski, W. and Morris, B. 25 Pollut. Eng 12 01-Apr-80 58-59

Typical Pipe Flow Problems Programmed for Limited Capability Calculators Marinos, A.T. 25 Water Sewage Works 127R 30-Jun-80 86-87

Programs for :1 Sewer Pipe Design, 2 Pipe culvert analysis, 3 Pipe culvert design, 4 Trapezoidal open channel design, 5 Trinagular open channel design, 6 Pipe network design 7 Sewer Pipe velocity and flow Waldo, J. 19C Water Eng Management 128R 30-Apr-81 52-55

Calculator Program aids lagoon design Palmenburg, R.E. 33C Water Eng Management 128 01-Nov-81 49-52 7

Standard Deviation program combines recurring data Nelson, R. 25 Electronics 49 09-Dec-76 115

Determining the significance of an observed F using a pocket calculator Kaemmerer, W.K. 25 Behav. Res Methods Instrum. 9 01-Jun-77 557-558

Moving Averages and ranges Claudy, N.H. 25 Tappi 63 01-Feb-80 177-178

Pocket Calculator Program for Least Square Fitting of Data with Variable Precision Picot, A. 25 Am. J. Phys. 48 01-Apr-80 302-303

Pocket Calculator Program: Welch's vs Statistic Jederberg, W and Gildengorin, V. 29C Comput. Biol. Med. 11 No 3 1981 167-169

A shortcut technique for finding H₂SO₄ concentration Davidson, C.M. 11C CE 96 01-Mar-89 185

Calculator Design of multistage evaporators Esplugas, S and Mata, J. 67/97 CE 90 07-Feb-83 59-61

Curve fitting via orthogonal polynomials Clare, B.W. 41 CE 90 18-Apr-83 85-88

Estimate Air Cooler size Shaikh, N.M. 41 CE 90 12-Dec-83 65-68

Evaluating separation processes Ingels, R.M. 67/97 CE 90 30-May-83 73-75

Program for evaluation of shell-and tube heat exchangers Crane, R and Gregg, R. 67/97 CE 91 25-Jul-83 76-80

Calculator program for new-controller tuning Jutan, A and Rodriguez, E. 67/97 CE 92 3/9/1984 69-73

- Two programs for multicomponent distillation Ngai, I 41 CE 92 20-Aug-84 145-149 8
- Simulating Heat Exchanger Performance Chang, S.C. 41 CE 92 02-Apr-84 81-88 9
- Use your calculator as an event counter Thomas, W.B. 11C CE 92 17-Sep-84 130 10
- Programs for fixed bed sorption Tan, H.K.S. 41 CE 92 24-Dec-84 57-61
- Program calculates venturi scrubber efficiency Ollero, P. 41 CE 92 28-May-84 103-105
- Program calculates losses for counter current decantation Eberts, D.H. 67/97 CE 92 12-Nov-84 121-124
- Packed Column design on a pocket calculator Hixson, T.J. 41 CE 92 06-Feb-84 95-98
- Absorber/stripper design with a programmable calculator Hile, L. 41 CE 93 29-Apr-85 53-56
- Analyse Cross Flow heat exchangers Turton, R. and Frederick, N. 41 CE 94 07-Jul-86
- Calculator Program for Finding Values of Physical Properties Wooler 41 CE 94 31-Mar-86 106-118 11
- Fourier Analysis and Synthesis with a Pocket Calculator Schmidt, S.A. 25 Am. J. phys. 45 01-Jan-77 79-82
- Program finds equation roots Pierce, F.J. 25 Mach. Des. 49 07-Apr-77 87-88
- Adding Dimensions in English units Zaremba, M.J. 25 CE 84 21-Nov-77 222
- A Short Program for Simpson's or Gazdar's rule-Integration on Handheld Programmable Calculators Gazdar, A.S. 25 Two-Year College Maths. J. 9 01-Jun-78 182-185
- Tracking down equation roots Reubens, A. 55 Mach. Des. 50 10-Aug-78 120
- Prime Numbers on the HP-19C Aslan, W. 19C BYTE 5 01-Oct-80 54-58
- Equilibrium Calculations with a Pocket Calculator Woicik, J.F. 65 CE 83 16-Aug-76 89-93 12
- Determining ideal stages on a pocket calculator Tan, H. 25 CE 84 24-Oct-77 158
- H- factor integration program for HP-65 Calculator Tasman, J.E. 65 Tappi 61 01-May-78 114-115
- Another approach - Amount of Inorganic Material in the Kraft Liquor Cycle Samuels, R.M. 34C Tappi 63 01-Nov-80 145-146
- Determine System Head Curves with a Programmable Calculator Dancs, L. 25 Water Sewage Works 124 01-Aug-77 61-63
- Maximizing Economy in a Multiple Effect Evaporator Beaujean, J.A. 25 Tappi 124 01-May-78 113-114
- More on calculating H-factors for Batch Digesters Samuels, R.M. 34C Tappi 64 01-Mar-81 179-180
- Heat recovery in Counterflow Heat Exchangers: An adaptation Samuels, R.M. 34C Tappi 64 01-May-81 131-132
- Molecular Calculations for Microanalysis on a Programmable Calculator: 1: Experimental Percentage Composition Corliss, J.M. 65 Microchem. J. 21 No 4 Dec 1976 458-465
- Simple methods for calculating reaction equilibria in single reaction or single moiety networks: 1: One Reaction Equilibrium - Stage 1; 2: One Reaction Equilibrium - Stage 2 Park, D.J.M. 25 Comput. Programs. Biomed. 6 01-Dec-

76 263-268

Use of a programmable pocket calculator in processing amino acid analysis data Buchanan, J.H. 25 J. Chromatog. 137 No 2 1977 475-480

Rapid Rigorous Computation of Modulation transfer Function on a Pocket Calculator Ronai, P.M. and Kirch, D.L. 65 J. Nucl Med. 18 No 6 June 1977 579-583

Crystallographic Computations on the Pocket Calculator: 1. Program to calculate d-spacings for a triclinic unit cell , 2: Dhkl spacings for a monoclinic cell, 3: To calculate bond distances for a triclinic unit cell, 4 to calculate torsion angles MacKay, A.L. 25 J. Crystal. Mol. Struct. 9 No 4 Dec 1979 223-231

HP-25 program makes fast cost estimates Barocio, J. 25 Electronics 52 02-Aug-79 138

Washer efficiency control by the use of the displacement ratio Lopes, C.H. 25 Tappi 62 01-Sep-79 115-116

Programmable Calculator simplify balancing of rotating equipment Scheneller, S.G. 33E Power 123 01-Dec-79 70-71

Finding Pipe intersections Andrako, T.E. Mach. Des. 52 07-Feb-80 113

Cutting Pipe intersections Becker, E.B. Mach. Des. 52 10-Apr-80 133-134

Calc program finds effective flow area Goldberg, U. 33E Mach. Des. 52 09-Oct-80 216

Calc Program finds gas pressure drop (pressure drop and pipe length) Bailey, B.M. 29C Mach. Des. 53 08-Jan-81 162-164

Estimate of economic pipe size Samuels, R.M. 34C Tappi 64 01-Jun-81 127-128

Solve valve sizing and noise calculations on site in minutes 1:Sizing calculation - gas (scfh) , 2 Sound level calculations - standard valves ($P_1/P_2 > 2.8$) Hynes, K. 65 Instrument Control Syst 49 01-Mar-76 43-47

Use of a programmable calculator to determine thermocouple temperatures Delinger, W.G. 25 Solar Energy 20 01-Apr-78 359-360

Solving Instrumentation Problems with a programmable calculator: 1: Control Valve sizing (gas) 2: Orifice sizing Buckley, P.S. and Mariam, P.L. 65 Instrum. Technol. 22 01-Feb-79 31-37

Program for Discounted Cashflow Return on Investment Wild, N.H. 25 CE 84 09-May-77 137-142

Hand held calculator augments optimized drilling programs- pt 5- a way to predict abnormal pressure Siegel, B 67/97 PEI 01-Sep-81 81-84

Hand held calculator program yields mooring line forces Mes, M.J. 67/97 PEI 6 01-May-81 28-37

HP 41 calculates Dykstra-Parsons permeability Bixler,R 41 PEI 01-Jul-83 58-60

Hand held calculator assists in directional drilling control- part 3 Marsh 41 PEI 01-Jan-83 48-56

Hand held calculator assists in directional drilling control- part 1 Marsh 41 PEI 01-Jul-82 79-88

Hand held calculator assists in directional drilling control- part 2 Marsh 41 PEI 01-Sep-82 82-88

Calculator helps to determine cash flow - present worth Yager, J.W. 41 PEI 01-Nov-83 76-82

HP41 helps predict oil production Bixler,R 41 PEI 01-Apr-82 80-82

- Hand held calculator optimizes drilling programs -pt 8- real time hydraulics Siegel, B 41 PEI 01-May-82 136-143
- Hand held calculator augments optimized drilling programs -part 7- solids analysis Siegel, B 41 PEI 01-Mar-82 130-140
- New procedure determines drilling fluid surge pressures Broussard, M.L. 67/97 PEI 01-Sep-82 56-60
- HP41CV program eases IPR test analysis Fox, M.J 41 OGJ 82 09-Jul-84 108-111
- Hand held calculator programs determine natural gas physical properties Ajitsaria, N.K. 67 OGJ 81 06-Jun-83 69-74
- Program predicts 2 phase pressure gradients Jacks, D.C & Hill, A.D. 41 OGJ 81 28-Nov-83 126-129
- Two calculator programs for Hazen-Williams pipeline equation Bartos, B.B. 11C OGJ 30-Apr-84 56-60
- Calculator program optimizes bit weight, rotary speed, reducing drilling cost Simpson, M.R. 41 OGJ 82 22-Apr-84 71-77
- Program for HP-67 calculator determines Maxwell Bonnell vapour pressure correlation Arnold, V.E. 67 OGJ 82 23-Apr-84 84-89
- Calculator program aids sucker rod systems design and optimization Engineer, R.& Knight, R. & Davis , C 41 OGJ 81 15-Aug-83 56-63
- Program calculates radius of curvature coordinates for straight deviated holes Oyler, D.C. OGJ 81 24-Oct-83 100-108
- Correlation eases absorber-equilibrium line calculations for TEG-natural gas dehydration Behr, W.R. 41 OGJ 81 07-Nov-83 96-98
- Program assists steam drive project Mendez AA 41 OGJ 82 27-Aug-84 94-98
- Calculator program aids cathodic protection design Tran, V.B 41 OGJ 82 06-Aug-84 69-72
- Calculator program helps set offshore design criteria Tran, V.B 41 OGJ 81 10-Jan-83 91-94
- Use of hand held calculator program aids analysis of subsea pipeline design Tran, V.B 41 OGJ 81 14-Feb-83 94-102
- Equations, program aid gas-flow calculations Ajitsaria, N.K. 67 OGJ 79 09-Feb-82 138-145
- 3P calculator program for multiphase flow analysis in pipelines, casing and tubes Hein, M 41 OGJ 79 16-Aug-82 148-152
- Calculator program gives pressure-vessel characteristics Scott, L 41 OGJ 83 04-Feb-85 60-65
- Program gives design data for multiple orifice system to avoid cavitation Goksem, P.G. & Patel, I.D 41 OGJ 83 15-Apr-85 56-62
- Routine for computer or calculator solves gas-flow problems Stannard, J.H. 41 OGJ 83 15-Apr-85 62-65
- Program speeds calculation for binary multicomponent viscosity blending Denchfield, T.D.& Pedall. R.F 67 OGJ 80 13-Sep-82 74-81
- Hand held calculator program gives economic evaluation Jones, R. & Maxwell, R.L, 41 OGJ 80 13-Sep-82 102-108
- Calculator program for rupture disk design Phadke, P.S. 97 HP 62 01-Dec-83 34B-34L

Friction loss in piping Patel, V.K. 41 HP 63 01-Jan-84 100-102

Shortcut program eases gas calculations Dutcher, W.G. 41 HP 61 01-Dec-82 81-84

Calculator Program analyses Insulated pipe Barritt, S.L. 67/97 Heat/Piping/Air Cond. 50 01-Mar-78 65-70

Notes

1 Also see CE 10/9/1979 page 5

2 Also see CE 14/6/1982 page 5

3 Also see Tappi 3/1979 page 109

4 Also see Tappi 11/1980 page 144

5 Correction 11/1/82 page 5

6 Also see BYTE Vol 4 Mar 1979 page 196-204

7 Also see Water Eng Management Vol 129 Feb 1982 page 66

8 Correction 29th October 1984 p 5

9 Correction 25th June 1984 p5

10 Correction 29/10/84 page 5

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

EE calculations, some not obvious

Posted by [Sam Levy](#) on 28 Mar 2007, 10:17 p.m.

HEWLETT PACKARD CALCULATIONS I used HP RPN pocket calculators for electronic solutions, and I wanted to find a fast way to solve common problems using as few keystrokes as possible, and getting maximum information during the process. This method uses the repeating t register. Formulas now in use were derived for pen and pencil mathematics, totally outmoded by the calculator. This is the keystroke procedure for quick solutions. Resistive Dividers; when calculating from known resistors, enter the lower resistor repeatedly. Key in the top resistor, and add, the result is the total resistance, then divide, the result is the fractional output voltage of the divider. Then multiply and subtract to see the output resistance from the divider tap. To design such a network, key in the output resistance desired, and enter repeatedly. Key in the division fraction, and divide to see the upper resistance. Divide again to restore the fraction, key 1, subtract, change sign, divide, and see the lower resistor. The relation between the upper resistor and the output resistance is the fraction of the divider. 1 minus the fraction is the relation between the output resistance and the lower resistance. Given the fraction and either resistor, the other value may be found. I have used this method to find the added end resistors for setting particular potentiometer control ranges. The difference between the two potentiometer fractions desired at the potentiometer ends, relates the potentiometer resistance to the total resistance. Thus a particular value of potentiometer may be used to obtain the desired range of fractions. EXAMPLE: It was desired to use a 1K potentiometer to adjust a power supply to vary between 24 and 16 volts. The IC regulated at 4.75 volts from the potentiometer. It is desired to calculate the end resistors needed to set the correct range. $4.75/24 = 0.2$ store 1. $4.75/16 = 0.3$ store 2. Recall 1, subtract, see 0.1, key 1000 swap X&Y, divide, see 10105, the total resistance. Enter several times. Recall 1, multiply, see 2000, the resistance when the pot is at the lower extreme, add 2000 ohms to the lower end of the pot. Cl X, recall 2, multiply, see 3000, this is the sum of the pot resistance and the pot lower end resistor. The upper end resistor must be 10205 minus 3000 = 7105. Some compromise values may be used. Transistor bias; I was solving for a transistor bias divider, and wondered how to account for the base current drop in the divider. I thought to divide the divider output resistance by Beta and add it to the actual emitter resistor. To do this procedure, solve the bias divider resistors, storing the divider fraction. With R0 in the register key Beta, divide, and see the emitter resistor equivalent to be added. Key the actual emitter resistor and add, see the total emitter resistor, accounting for the bias source drop. Key the source voltage; recall the fraction, multiply and see the divider voltage. Key 0.7, subtract, see the emitter voltage. Interchange X&Y and divide, to see the emitter current: check by multiplying by the collector resistor, to see the collector resistor drop.

Summing network; The resistive divider can be extended to a more general case. I needed to design a telemetry output that combined several signals, to a common output, having a 10 K output resistance. The signal of interest was a +/- 12 volt VCO control voltage. The telemetry output was to be 0-5 volts, using the range 0.5 to 4.5 volts normally. The centering voltage was 2.5 volts from a regulated 12 volts supply. A third resistor to ground gave the desired output resistance. I found a general solution. Each resistor is determined by the fraction of its input voltage contributed to the output sum, thus the telemetry signal was to output 1/6th of its signal. The resistor was then 10K times 6 or 60K. The centering voltage was 2.5 volts derived from +12, a ratio of 4.8 times, 48K. The third resistor was found by subtracting the other two fractions from 1: $1 - 1/6 - 1/4.8$, a fraction of 0.625 and dividing the output resistance to get a resistance of 16K. Some compromise values were used, which proved satisfactory. RC frequency corners; the process may be simplified as $2\pi FRC = 1$. The known values are multiplied, and the product inverted to give the missing element. For constant frequency solutions enter $2\pi F$ repeatedly. Tuned circuits; here intermediate results in the solution give other

useful information. Key in L, then C, and store C in a register, divide and take the square root; this is the impedance of the resonant elements. Recall the capacitance value and multiply, then multiply by 2Pi and invert to see the frequency in Hertz. This is the frequency corner formula using R0 as the R value. Q if known may be used to compute the resonant impedance, or the Q may be found by dividing the parallel circuit damping resistance by the impedance, or the impedance by the series circuit resistance. The square root of L/C is also the output resistance of a simple LC filter, such as a power supply, and can be used to estimate transient voltages from current changes. Resonant values; When using a constant frequency, resonant values may be calculated quickly. Key in 2PiF and enter it repeatedly. Then key in either L or C , multiply twice, and invert to see the other resonant circuit element. When 2PiFL is shown, it is the reactance of the circuit values. This procedure gives quick trials of various circuit values, give impedance values for filter and by-pass elements, and value choices for a given Q. Ohms law; this may seem simple, but can be used as a rapid verification of dissipation and the voltage and current relation. Remember the little circles of voltage E divided by the IR product. Key in the voltage repeatedly, and divide by either I or R to see the other value, as in E=IR. The power in a resistor can be found by keying in the resistor and dividing to get I, then multiplying to get EI. It can be used to solve W=EI, by keying in power repeatedly, and dividing to get either E or I. These methods allow such rapid solutions; they will increase your ability to investigate possibilities in your designs. Practice will allow fast and sure answers to a wide range of problems. Familiarity will allow you to extend these methods to your particular problems. I trained one tech to calculate time payments faster than they could be found in a book. E. Samuel Levy, 754 Temple St. San Diego 92106, 619-223-6292 designnut@cox.net

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[Return to the Index]
[Previous | Next]

Calculator Benchmark

Posted by [Xerxes](#) on 12 Feb 2007, 7:35 a.m.

Calculator Speed Benchmark using the N-Queens Problem

-		
-	5:00:00	MK-52 Keystroke / RPN
-		
-	4:45:00	MK-61 Keystroke / RPN
-		
-	1:50:00	TI-66 Keystroke
-		
-	1:37:50	HP-16C Keystroke / RPN
-		
-	1:19:10	HP-15C Keystroke / RPN
-		
-	1:13:10	HP-34C Keystroke / RPN
-		
-	1:03:50	HP-11C Keystroke / RPN
-		
-	58:20	TI-58C Keystroke
-		
-	56:15	SR-52 Keystroke
-		
-	55:52	HP-12C Keystroke / RPN
-		
-	53:47	FX-201P Keystroke / Fortran Style
-		
-	47:57	HP-67 Keystroke / RPN
-		
-	46:20	PC-1211 Basic
-		
-	43:40	TI-59 Keystroke
-		
-	42:33	PC-1212 Basic
-		
-	42:00	HP-29C Keystroke / RPN
-		
-	40:50	TI-66 Keystroke / Turbo x2.7
-		
-	25:16	HP-15C Keystroke / RPN / Turbo x3.1
-		
-	24:33	TI-59 Keystroke / Fast Mode x1.8
-		
-	23:24	HP-12C Platinum Keystroke / RPN
-		
-	20:00	FX-4500PA Formula / Array
-		
-	18:40	PC-1300S Keystroke / mini Fortran
-		
-	18:25	TI-88 Keystroke / Prototype
-		
-	17:58	HP-41CL Keystroke / RPN / TurboX Mode x1.0
-		
-	17:56	HP-41CX Keystroke / RPN
-		
-	17:50	EL-9000 Formula / Vector

-			
-	17:42	HP-41CV	Keystroke / RPN
-	17:40	FX-4500P	Formula / Array
-	17:35	HP-41C	Keystroke / RPN
-	17:04	HP-41CY	Keystroke / RPN / Turbo Off
-	16:13	PX-5600PV	Formula / Array
-	15:10	FX-702P	Basic
-	14:19	HP-38G	Formula / List
-	14:10	SRP-400G	Formula / List
-	12:12	HP-42S	Keystroke / RPN / Ver.C
-	11:55	FX-750P	Basic
-	11:34	HP-42S	Keystroke / RPN / Ver.A
-	10:49	PB-700	Basic
-	10:41	MK-85	Basic
-	10:20	PB-770	Basic
-	10:18	TI-80	Formula / List
-	10:18	FX-502P	Keystroke
-	9:50	EL-5030	Formula / Vector
-	9:47	SF-9350	Formula / Array / Scientific ROM Card ES-650
-	9:35	HP-28C	UserRPL
-	9:28	HP-41CL	Keystroke / RPN / Turbo2 Mode x1.9
-	9:08	VX-4	C / Unstructured / Bytecode
-	8:52	PB-2000C	C / Unstructured / Bytecode
-	8:29	PC-1250A	Basic
-	8:23	PC-1401	Basic
-	8:22	PC-1245	Basic
-	8:22	PC-1251	Basic
-	8:19	PC-1402	Basic
-	8:13	HP-41CY	Keystroke / RPN / Turbo On x2.1
-	8:00	FX-795P	Basic / DEFM Mode
-	7:45	FX-602P	Keystroke
-	7:45	PC-1365	Basic
-	7:40	PC-1360	Basic
-	7:36	FX-790P	Basic / DEFM Mode
-	7:25	SF-R10	Formula / Array / Scientific ROM Card ES-650
-	7:23	IHC-8000	Basic
-	7:22	V12	Keystroke / RPN
-	7:22	CCS28956	Keystroke / RPN

-			
-	7:21	PC-1280	Basic
-	7:17	PC-1460	Basic
-	7:15	AI-1000	Lisp / Bytecode
-	7:14	PC-1475	Basic
-	7:14	PC-1285	Basic
-	7:11	FN-1000	Keystroke / RPN
-	7:04	TPC-8300	Basic / DEFINT
-	7:00	FX-730P	Basic / DEFM Mode
-	6:55	PB-110	Basic
-	6:51	PC-1350	Basic
-	6:49	FX-720P	Basic
-	6:42	PC-1450	Basic
-	6:38	EL-9600	Formula / List
-	6:36	FX-5200P	Basic
-	6:36	PC-1445	Basic
-	6:35	PC-1440	Basic
-	6:30	PB-80	Basic
-	6:30	UC-2200	Basic
-	6:29	MK-90	Basic / Ver.1.0
-	6:29	MK-90	Basic / Ver.2.0
-	6:28	HP-28S	UserRPL
-	6:25	HP-17BII+	Equation Solver
-	6:24	PC-1262	Basic
-	6:19	PC-1403	Basic
-	6:11	HP-48S	UserRPL
-	6:08	HP-48SX	UserRPL
-	6:08	PB-220	Basic
-	6:07	HP-42S	Keystroke / RPN / Ver.C / Turbo x2.0
-	6:02	HP-42S	Keystroke / RPN / Ver.C / Fast Mode x2.0
-	5:59	TI-74 BASICALC	Learn Pascal ROM Module / Interpreter
-	5:56	FX-4000P	Formula / Array
-	5:48	HP-42S	Keystroke / RPN / Ver.A / Turbo x2.0
-	5:47	SRP-320G	Formula / Array
-	5:47	828	Formula / Array / SC-828
-	5:45	PB-120	Basic
-	5:44	HP-32SII	Keystroke / RPN
-	5:35	PC-1440	CAP-X Assembly / COMP-X Simulator

-			
-	5:05	FX-702P	Basic / Turbo x3.0
-	5:03	HP-40GS	Formula / List
-	4:52	PB-100	Basic
-	4:52	EL-9400	Formula / List
-	4:48	EL-9300	Formula / Vector
-	4:41	FX-700P	Basic
-	4:33	HP-49G	UserRPL
-	4:30	HP-71B	Keystroke / RPN / 41-Translator ROM Module HP-82490A
-	4:24	PC-2001	Basic
-	4:22	HP-32S	Keystroke / RPN
-	4:17	HP-35S	Keystroke / RPN
-	4:15	TI-86	Formula / List
-	4:08	HP-41CL	Keystroke / RPN / Turbo5 Mode x4.3
-	4:05	PB-2000C	C / Unstructured / Bytecode / Turbo x2.2
-	4:02	HP-48GX	UserRPL / Ver.P
-	3:59	TI-74 BASICALC	Learn Pascal ROM Module / Interpreter / Turbo x1.5
-	3:59	PC-1270	Basic
-	3:57	PC-1402	Basic / Turbo x2.1
-	3:55	VX-1	Basic
-	3:53	TI-95 PROCALC	Keystroke
-	3:52	FX-840P	Basic
-	3:52	FX-860PVC	CASL Assembly / COMET Simulator
-	3:51	FX-860PVC	Basic
-	3:51	FX-880P	Basic
-	3:50	FX-502P	Keystroke / Turbo x2.7
-	3:47	FX-5800P	Formula / Array
-	3:42	PC-1360	Basic / Turbo x2.1
-	3:40	PC-1445	CASL Assembly / COMET Simulator
-	3:39	FX-7700GH	Formula / Array
-	3:32	PB-1000	Basic
-	3:30	HP-42S	Keystroke / RPN / Ver.A / Turbo + Fast Mode x3.3
-	3:29	HP-17BII	Equation Solver
-	3:28	HP-42S	Keystroke / RPN / Ver.C / Turbo + Fast Mode x3.5
-	3:26	PC-1500A	Basic / @-Array
-	3:20	AI-1000	Lisp / Bytecode / Turbo x2.2
-	3:15	PB-2000C	Basic ROM Card OM-53B
-	3:15	DM-15CC	Keystroke / RPN

-			
-	3:14	FP-200	Basic
-	3:14	VX-3	Basic
-	3:11	FX-870P	Basic
-	3:11	PC-1100	Basic
-	3:10	HP-28S	UserRPL / Fast Mode x2.0
-	3:08	VX-4	Basic
-	3:00	PC-1150	Basic
-	2:52	PC-1430	Basic
-	2:50	PB-2000C	Prolog ROM Card OM-51P / Bytecode
-	2:49	HP-35S	Keystroke / RPN / Turbo x1.5
-	2:45	PC-1248	Basic
-	2:45	VX-3	CASL Assembly / COMET Simulator
-	2:43	VX-4	CASL Assembly / COMET Simulator
-	2:43	FX-870P	CASL Assembly / COMET Simulator
-	2:43	TI-92+	Formula / List / Ver.2.04
-	2:41	TI-83+	Formula / List
-	2:40	FX-603P	Keystroke
-	2:37	FX-730P	Basic / DEFM Mode / Turbo x2.7
-	2:37	PC-1246	Basic
-	2:37	PC-1247	Basic
-	2:36	TI-73	Formula / List
-	2:35	TI-82	Formula / List
-	2:35	TI-95 PROCALC	Keystroke / Turbo x1.5
-	2:33	HP-71B	Basic / Ver.1BBBB
-	2:33	HP-71B	Basic / Ver.2CDCC
-	2:32	TI-89	Formula / List / HW1 / Ver.2.09
-	2:31	FX-7700GB	Formula / Array
-	2:29	CC-40	Basic
-	2:27	FX-8700G	Formula / Array
-	2:26	HP-41CL	Keystroke / RPN / Turbo10 Mode x7.4
-	2:23	FX-602P	Keystroke / Turbo x3.3
-	2:21	DM-12CC	Keystroke / RPN
-	2:20	ClassPad 330	Formula / List / OS 3.3
-	2:18	TI-74 BASICALC	Basic
-	2:16	FX-4850P	Formula / Array
-	2:15	FX-4800P	Formula / Array
-	2:15	FX-6300G	Formula / Array

-			
-	2:13	PC-1500A	Basic / @-Array / Turbo x1.5
-	2:11	HP-33S	Keystroke / RPN
-	2:10	FX-4000P	Formula / Array / Turbo x2.7
-	2:08	ClassPad 300+	Formula / List / OS 2.2
-	2:07	FX-8000G	Formula / Array
-	2:06	FX-8500G	Formula / Array
-	2:06	FX-6000G	Formula / Array
-	2:05	AFX-2.0+	Formula / List
-	2:05	PB-770	Basic / Turbo x5.0
-	2:01	FX-6500G	Formula / Array
-	2:00	FX-7000G	Formula / Array
-	2:00	MK-85	Basic / Fast Mode x5.3
-	1:59	FX-7400G+ GY	Formula / List
-	1:59	FX-7450G	Formula / List
-	1:57	FX-880P	Basic / Turbo x2.0 (1.228->2.456 MHz)
-	1:57	FX-7400G+	Formula / List
-	1:57	FX-7200G	Formula / Array
-	1:53	TI-85	Formula / List / Ver.10.0
-	1:51	FX-7000GB	Formula / Array
-	1:49	FX-7000GA	Formula / Array
-	1:49	OH-7000G	Formula / Array
-	1:48	TI-89	Formula / List / HW2
-	1:46	FX-9750G	Formula / List
-	1:45	FX-700P	Basic / Turbo x2.7
-	1:45	RM-9850G	Formula / List
-	1:45	CFX-9950GB+	Formula / List
-	1:44	TI-89 Titanium	Formula / List / HW3
-	1:44	TI-89 Titanium	Formula / List / HW4
-	1:44	CFX-9850GB+	Formula / List
-	1:44	CFX-9970G	Formula / List
-	1:43	FX-9750G+	Formula / List
-	1:39	PC-1600	Basic / @-Array
-	1:39	FC-1000	Formula / Cash Flow List
-	1:37	PB-1000	Basic / Turbo x2.2
-	1:35	HP-41CL	Keystroke / RPN / Turbo20 Mode x11.3
-	1:34	FX-CP400	Formula / List
-	1:32	TI-74 BASICALC	Basic / Turbo x1.5

-			
-	1:32	828	Formula / Array / SC-1376
-	1:31	ATC-139	Formula / Array / SC-1418
-	1:31	ClassPad 300+	Formula / List / OS 2.2 / Fast Mode x1.4 (29.5->59 MHz)
-	1:30	HP-49G+	UserRPL / Ver.2.15
-	1:30	HP-50G	UserRPL
-	1:30	FX-7500G	Formula / Array
-	1:30	PB-2000C	Basic ROM Card OM-53B / Turbo x2.2
-	1:29	TI-84+ Pocket SE	Formula / List / Ver.2.55MP
-	1:29	TI-Nspire	Formula / List / TI-84+ Keypad / OS 3.0
-	1:26	PC-1248	Basic / Turbo x1.9 (1.024->2.0 MHz)
-	1:24	PC-G801	Basic
-	1:22	IQ-7100	Basic ROM Card IQ-870
-	1:22	IQ-7720	Basic ROM Card IQ-770
-	1:21	PC-E220	Basic
-	1:20	IQ-7520	Basic / Scientific ROM Card IQ-8B03
-	1:20	FX-603P	Keystroke / Turbo x2.0 (1.228->2.456 MHz)
-	1:19	FX-790P	CAP-X Assembly / COMP-X Simulator
-	1:19	PC-E500S	Basic
-	1:18	PC-1490UII	Basic
-	1:18	PB-2000C	Prolog ROM Card OM-51P / Bytecode / Turbo x2.2
-	1:14	CFX-9800G	Formula / Array
-	1:14	HP-19BII	Equation Solver
-	1:13	TI-Nspire	Formula / List / TI-84+ Keypad / OS 1.2
-	1:10	TI-Nspire	Formula / List / TI-84+ Keypad / OS 1.3
-	1:08	TI-84+	Formula / List
-	1:08	FX-7300G	Formula / Array
-	1:07	HP-50G	UserRPL / Fast Mode x1.3 (75->203 MHz)
-	1:05	HP-41CL	Keystroke / RPN / Turbo50 Mode x16.6
-	1:04	C64	Basic
-	1:04	PC-G850V	CASL Assembly / COMET Simulator
-	1:04	PC-G850VS	CASL Assembly / COMET Simulator
-	1:02	TI-83+ SE	Formula / List
-	1:02	FX-9700GH	Formula / Array
-	1:00	FX-9700GE	Formula / Array
-	59.5	IQ-8500	Basic / Scientific ROM Card IQ-8B03
-	58	Z-1	Basic
-	57.5	IQ-9200	Basic / Scientific ROM Card IQ-8B03

-			
-	56.9	Z-1GRA	Basic
-	56.8	FX-890P	Basic
-	56	FX-840P	CASL Assembly / COMET Simulator
-	55.0	FX-7000G	Formula / Array / Turbo x2.2 (0.91->2.0 MHz)
-	54	X-07	Basic / DEFINT
-	54.0	CFX-9850GB+	Formula / List / Turbo x1.9 (4.3->8.0 MHz)
-	53.1	TI-89	Formula / List / HW2 / Turbo x2.0
-	53.0	IQ-8900	Basic / Scientific ROM Card IQ-8B03
-	49.6	TI-81	Formula / Statistic List / Ver.1.6K
-	47	HP-75C	Basic / Fast Integer / Ver.aaaaaa
-	46.3	HP-71B	Forth / 41-Translator ROM Module HP-82490A
-	44	HP-75D	Basic / Fast Integer / Ver.aaaaaa
-	43.8	FX-7000GA	Formula / Array / Turbo x2.5 (1.6->4.0 MHz)
-	42.7	EL-9900	Formula / List
-	42.5	FX-6300G	Formula / Array / Turbo x3.2 (0.8->2.6 MHz)
-	40	FX-CG20	Formula / Matrix
-	38.3	PC-E500S	Basic / Turbo x2.1
-	38.3	PC-G850V	Basic
-	37.7	PC-G850VS	Basic
-	37.2	PC-G850S	Basic
-	36.2	PC-G850	Basic
-	35.9	PC-E220	Basic / Turbo x2.3
-	35.2	HP-48GX	SysRPL / Ver.R
-	34.2	TI-85	Formula / List / Ver.10.0 / Turbo x3.3
-	32.3	FX-9860G II	Formula / Matrix / OS 2.0
-	32	HP-75D	Basic / Fast Integer / Ver.dddddd
-	31.7	PB-1000	PB-Forth
-	30.0	FC-1000	Formula / Cash Flow List / Turbo x3.3 (2.0->6.6 MHz)
-	29.0	FX-9750G II	Formula / Matrix / OS 2.0
-	28.7	FX-9860G Slim	Formula / Matrix / OS 2.0
-	28	HP-15C LE	Keystroke / RPN
-	27.3	FX-7500G	Formula / Array / Turbo x3.3 (2.0->6.6 MHz)
-	23.5	Z-1GRA	CASL Assembly / COMET Simulator
-	23.5	FX-890P	CASL Assembly / COMET Simulator
-	21.3	HP-12C ARM	Keystroke / RPN
-	19.5	TI-81	Formula / Statistic List / Ver.1.6K / Turbo x2.5
-	18.7	HP-200LX	GW-Basic 3.22 / DEFINT

-			
-	17.5	FX-9860G II	Formula / Matrix / OS 2.0 / Fast Mode x1.8 (29.5->118 MHz)
-	17.3	FX-9860G SD	Formula / Matrix / OS 1.0
-	17.0	FX-9860G Slim	Formula / Matrix / OS 1.0
-	15.3	MK-161	Keystroke / RPN / Ver.1.03
-	14.9	MK-152	Keystroke / RPN / Ver.1.15
-	14.7	MK-152	Keystroke / RPN / Ver.1.05
-	14.6	PB-1000	PB-Forth / Turbo x2.2
-	12.9	Z-1GRA	C / Unstructured / Bytecode
-	12.9	FX-890P	C / Unstructured / Bytecode
-	11.0	HP-41CX	Mcode / NUT @ 0.36 MHz
-	10.80	HP-41CL	Mcode / TurboX Mode x1.0
-	9.82	Organiser II XP	OPL / Bytecode
-	9.31	HP-30B	Keystroke / RPN
-	9.07	FX-9860G SD	Formula / Matrix / OS 1.0 / Fast Mode x1.9 (29.5->118 MHz)
-	8.80	FX-9860G Slim	Formula / Matrix / OS 1.0 / Fast Mode x1.9 (29.5->118 MHz)
-	8.3	TI-Nspire CAS	Formula / List / OS 1.2
-	7.5	TI-Nspire	Formula / List / OS 1.2
-	6.8	WP-34S	Keystroke / RPN / Real Mode / Ver.3.3
-	6.68	TG-202	Formula / Array
-	6.1	WP-34S	Keystroke / RPN / Integer Mode / Ver.3.3
-	5.40	HP-41CL	Mcode / Turbo2 Mode x2.0
-	4.28	PC-G850V	C / Unstructured / Bytecode
-	4.25	PC-G850VS	C / Unstructured / Bytecode
-	4.2	TI-Nspire CAS	Formula / List / OS 1.3
-	3.9	TI-Nspire	Formula / List / OS 1.3
-	3.37	HP-48GX	HP48XGCC / Structured / Cross Compiler
-	3.30	HP-200LX	QBasic 1.1 / DEFINT / Bytecode
-	2.92	Series 3a	OPL / Bytecode
-	2.45	HP-49G	HP Pascal Studio 2.1 / Cross Compiler
-	2.3	TI-Nspire CAS	Formula / List / OS 3.0
-	2.3	WP-34S	Keystroke / RPN / Real Mode / Ver.2
-	2.22	HP-200LX	Power Basic 3.2 / DEFINT / Native Compiler
-	2.17	HP-41CL	Mcode / Turbo5 Mode x5.0
-	2.1	TI-Nspire	Formula / List / OS 3.0
-	2.1	WP-34S	Keystroke / RPN / Integer Mode / Ver.2
-	1.82	HP-200LX	Turbo Basic 1.1 / DEFINT / Native Compiler
-	1.32	HP-39GII	Formula / List

-			
-	1.27	PB-2000C	DL-Pascal ROM Card 1.2 / Native Compiler
-	1.22	MK-85	Assembly / KA1013BM1 @ 0.36 MHz
-	1.10	ClassPad 300+	CPLua 0.8 / Bytecode
-	1.09	HP-41CL	Mcode / Turbo10 Mode x9.9
-	1.06	PC-1402	Assembly / SC61860 @ 0.576 MHz
-	0.983	Series 3mx	OPL / Bytecode
-	0.900	HP-200LX	QuickBasic 4.5 / DEFINT / Bytecode
-	0.895	HP-50G	HP Pascal Studio 2.1 / Cross Compiler
-	0.796	PC-1360	Assembly / SC61860 @ 0.768 MHz
-	0.770	ClassPad 300+	CPLua 0.8 / Bytecode / Fast Mode x1.4 (29.5->59 MHz)
-	0.622	HP-50G	HP Pascal Studio 2.1 / Cross Compiler / Fast Mode x1.5 (75-
>203 MHz)			
-	0.586	PB-2000C	DL-Pascal ROM Card 1.2 / Native Compiler / Turbo x2.2
-	0.55	HP-41CL	Mcode / Turbo20 Mode x19.6
-	0.385	PC-1360	Assembly / SC61860 @ 1.6 MHz / Turbo x2.1
-	0.346	HP-Prime	Formula / List
-	0.344	PB-1000	Assembly / HD61700 @ 0.91 MHz
-	0.336	HP-48GX	Assembly / Saturn @ 3.7-4.0 MHz
-	0.321	FX-8000G	Assembly / µPD1007 @ 0.91 MHz
-	0.312	PB-2000C	Assembly / HD61700 @ 0.91 MHz
-	0.305	PC-1500A	Assembly / LH5801 @ 1.3 MHz
-	0.245	SF-9350	Assembly / HD62076 @ 2.01 MHz / Long Memory Access Mode
-	0.245	SF-9700	Assembly / µPD3055 @ 2.01 MHz / Long Memory Access Mode
-	0.243	HP-50G	HPLua 0.4 / Bytecode
-	0.240	Series 5mx	OPL / Bytecode
-	0.23	HP-41CL	Mcode / Turbo50 Mode x47.0
-	0.220	MK-85	Assembly / KA1013BM1 @ 2.0 MHz / Fast Mode x5.5
-	0.198	PC-1500A	Assembly / LH5801 @ 2.0 MHz / Turbo x1.5
-	0.196	FX-7500G	Assembly / µPD1037 @ 2.0 MHz
-	0.195	SF-R20	Assembly / HD62076 @ 2.01 MHz / Short Memory Access Mode
-	0.176	MK-90	Assembly / K588 @ -.- MHz / Asynchronous Logic
-	0.158	PB-1000	Assembly / HD61700 @ 2.0 MHz / Turbo x2.2
-	0.144	PB-2000C	Assembly / HD61700 @ 2.0 MHz / Turbo x2.2
-	0.120	TI-74 BASICALC	Assembly / TMS70C46 @ 2.0 MHz
-	0.120	TI-95 PROCALC	Assembly / TMS70C46 @ 2.0 MHz
-	0.114	PC-E500S	Assembly / SC62015 @ 2.3 MHz
-	0.103	HP-200LX	Turbo Pascal 5.5 / Native Compiler
-			

-	0.100	C64	Assembly / 6510 @ 1.0 MHz
-	0.0975	HP-200LX	QuickBasic 4.5 / DEFINIT / Native Compiler
-	0.0950	CC-40	Assembly / TMS70C20 @ 2.5 MHz
-	0.0907	PC-E220	Assembly / SC7852 @ 3.58 MHz
-	0.0886	HP-200LX	Turbo C 2.01 / Unstructured / Native Compiler
-	0.0800	TI-74 BASICALC	Assembly / TMS70C46 @ 3.0 MHz / Turbo x1.5
-	0.0800	TI-95 PROCALC	Assembly / TMS70C46 @ 3.0 MHz / Turbo x1.5
-	0.0683	Z-1GRA	Assembly / 80L188EB @ 3.68 MHz
-	0.0683	FX-890P	Assembly / 80L188EB @ 3.68 MHz
-	0.0593	FX-7500G	Assembly / µPD1037 @ 6.6 MHz / Turbo x3.3
-	0.0552	PC-E500S	Assembly / SC62015 @ 4.8 MHz / Turbo x2.1
-	0.0504	TI-Nspire CAS	Lua / Bytecode / OS 3.1
-	0.0445	TI-Nspire CX CAS	Lua / Bytecode / OS 3.1
-	0.0400	PC-E220	Assembly / SC7852 @ 8.0 MHz / Turbo x2.3
-	0.0359	PC-G850S	Assembly / Z80 @ 8.0 MHz
-	0.0354	PC-G850V	Assembly / Z80 @ 8.0 MHz
-	0.0353	PC-G850VS	Assembly / Z80 @ 8.0 MHz
-	0.0295	HP-200LX	Assembly / 80186 @ 7.91 MHz
-	0.0255	TI-89	TIGCC 4.1.2 / Structured / Cross Compiler / HW2
-	0.0188	TI-83+ SE	Assembly / Z80 @ 15 MHz
-	0.0183	TI-89	Assembly / 68000 @ 12 MHz / HW2
-	0.00900	TI-89	Assembly / 68000 @ 24 MHz / HW2 / Turbo x2.0
-	0.00613	HP-50G	HPGCC 3.0 / RegVars / Cross Compiler / Slow Mode x12.7 (75->6.0 MHz)
-	0.00524	HP-50G	Assembly / ARM9 @ 6.0 MHz / Slow Mode x13
-	0.00248	FX-9860G SD	Casio C SDK 1.0 / Unstructured / Cross Compiler
-	0.00123	FX-9860G SD	Assembly / SH-3 @ 29.5 MHz
-	0.00123	FX-9860G II	Assembly / SH-3 @ 29.5 MHz
-	0.000691	FX-9860G SD	Casio C SDK 1.0 / Unstructured / Cross Compiler / Fast Mode x3.6 (29.5->118 MHz)
-	0.000484	HP-50G	HPGCC 3.0 / RegVars / Cross Compiler
-	0.000404	HP-50G	Assembly / ARM9 @ 75 MHz
-	0.000315	FX-9860G SD	Assembly / SH-3 @ 118 MHz / Fast Mode x3.9
-	0.000192	HP-50G	HPGCC 3.0 / RegVars / Cross Compiler / Fast Mode x2.5 (75->192 MHz)
-	0.000150	HP-50G	Assembly / ARM9 @ 192 MHz / Fast Mode x2.7

(391)

=====

Simple test used in case of incapability of executing n-queens (loops/min)

-		
-	30	TI-62 LBL 0 RCL 1 + 1 = ST0 1 RCL 1 x>t GT0 0
-	48	P-50 RCL + 1 = ST0 RCL SKP GOTO 00 R/S
-	55	TI-65 LBL A RCL 1 + 1 = ST0 1 RCL 1 x>m 0 GT0 A
-	57	TI-66 LBL A RCL 01 + 1 = ST0 01 RCL 01 x>=t A (As reference)
-	58	TI-57 II LBL 0 RCL 1 + 1 = ST0 1 RCL 1 x>=t GT0 0
-	80	TI-57 LBL 0 RCL 1 + 1 = ST0 1 RCL 1 x>=t GT0 0
-	85	PR-56D-NC RCL 1 + 1 = ST0 1 RCL 1 SKIP GOTO 00
-	95	SRP-60 RCL 1 + 1 = ST0 1 RCL 1 x>0 -9
-	97	PR-100 MR 1 + 1 = M 1 MR 1 SKIP GOTO 00
-	150	Enterprise RCL 1 + 1 = ST0 1 RCL 1 GIN 01
-	152	SR-56 RCL 1 + 1 = ST0 1 RCL 1 x>=t 00
-	159	HP-33C RCL 1 1 + ST0 1 RCL 1 x>0 GT0 01
-	330	Cambridge RCL + 1 = ST0 RCL GIN 00
-	350	TI-67 lbl 0 A+1->A if A>0 goto 0
-	355	FX-180P Kout 1 + 1 = Kin 1 Kout 1 x>0
-	360	FX-3600P Kout 1 + 1 = Kin 1 Kout 1 x>0
-	360	FX-50F Kout 1 + 1 = Kin 1 Kout 1 x>0
-	365	FX-3500P Kout 1 + 1 = Kin 1 Kout 1 x>0
-	380	FX-180PA Kout 1 + 1 = Kin 1 Kout 1 x>0
-	390	FX-3400P Kout 1 + 1 = Kin 1 Kout 1 x>0
-	450	FX-190 MR + 1 = Min MR x>0
-	635	SRP-350 Lbl 0: A=A+1 IF(A<0)THEN{} GOTO 0 (Buggy GOTO)
-	825	FX-3650P Lbl 0 A+1->A A>0=>Goto 0
-	855	FX-3800P Kout 1 + 1 = Kin 1 Kout 1 x>0
-	885	FX-200P Kout 1 + 1 = Kin 1 Kout 1 x>0
-	980	SRP-325G Lbl 0: A=A+1 IF(A<0)THEN{} GOTO 0 (Buggy GOTO)
-	1150	SR-12PR RCL 1 + 1 = ST0 1 RCL 1 x>0 -9
-	1160	SRP-175 RCL 1 + 1 = ST0 1 RCL 1 x>0 -9
-	1170	F-800P RCL 1 + 1 = ST0 1 RCL 1 x>0 -9
-	1190	EL-512H \-> A+1=>A A>0=Y=>[<- /]
-	1240	AC-692 RCL 1 + 1 = ST0 1 RCL 1 x>0 -9
-	1260	EL-5020 A+1=>A A x>M
-	1260	FX-3600PV Kout 1 + 1 = Kin 1 Kout 1 x>0
-	1290	PC-1201 LBL 0 RM 1 + 1 = x->M 1 RM 1 x<0 0 HLT
-	1540	FX-180PV Kout 1 + 1 = Kin 1 Kout 1 x>0
-	1720	F-700 Kout 1 + 1 = Kin 1 Kout 1 x>0
-	1720	SC-105 Kout 1 + 1 = Kin 1 Kout 1 x>0
-	1830	HP-20S LBL 0 RCL 1 + 1 = ST0 1 RCL 1 x<=y? GT0 0
-	1900	EL-5050 \-> A+1=>A A>0=Y=>[<- /]

```
-
- 2050 FC-200      Lbl 0 A+1->A A>0=>Goto 0
-
- 2280 EL-5120     LABEL 0 A=A+1 IF A>0 GOTO 0
-
- 2610 FX-50FH     Lbl 0 A+1->A A>0=>Goto 0
-
- 2710 FX-50F+     Lbl 0 A+1->A A>0=>Goto 0
-
- 2710 SC-3610     Kout 1 + 1 = Kin 1 Kout 1 x>0
-
- 2740 AC-3270     Kout 1 + 1 = Kin 1 Kout 1 x>0
-
- 3340 FX-3900P    Lbl 0 A+1->A 0<A=>Goto 0
-
- 3820 FX-3900PV   Kout 1 + 1 = Kin 1 Kout 1 x>0
-
- 7810 EL-5250     LABEL 0 A=A+1 IF A>0 GOTO 0
-
- 15450 SC-185     Lbl 0 A+1->A A>0=>Goto 0 (Ver.2)
```

(49)

=====

- CASIO
- - AI-1000
 - AFX-2.0+
 - CFX-9800G
 - CFX-9850GB+
 - CFX-9950GB+
 - CFX-9970G
 - ClassPad 300+
 - ClassPad 330
 - FC-200
 - FC-1000
 - FP-200
 - FX-50F
 - FX-50F+
 - FX-50FH
 - FX-180P
 - FX-180PA
 - FX-180PV
 - FX-190
 - FX-200P
 - FX-201P
 - FX-502P
 - FX-602P
 - FX-603P
 - FX-700P
 - FX-702P
 - FX-720P
 - FX-730P
 - FX-750P
 - FX-790P
 - FX-795P
 - FX-840P
 - FX-860PVC
 - FX-870P
 - FX-880P
 - FX-890P
 - FX-3400P
 - FX-3500P
 - FX-3600P
 - FX-3600PV
 - FX-3650P
 - FX-3800P
 - FX-3900P
 - FX-3900PV
 - FX-4000P
 - FX-4500P
 - FX-4500PA
 - FX-4800P
 - FX-4850P
 - FX-5200P
 - FX-5800P

FX-6000G
FX-6300G
FX-6500G
FX-7000G
FX-7000GA
FX-7000GB
FX-7200G
FX-7300G
FX-7400G+
FX-7400G+ GY
FX-7450G
FX-7500G
FX-7700GB
FX-7700GH
FX-8000G
FX-8500G
FX-8700G
FX-9700GE
FX-9700GH
FX-9750G
FX-9750G+
FX-9750G II
FX-9860G SD
FX-9860G Slim
FX-9860G II
FX-CG20
FX-CP400
OH-7000G
PB-80
PB-100
PB-110
PB-120
PB-220
PB-700
PB-770
PB-1000
PB-2000C
RM-9850G
SF-9350
SF-9700
SF-R10
SF-R20
VX-1
VX-3
VX-4
Z-1
Z-1GRA

SHARP

EL-512H
EL-5020
EL-5030
EL-5050
EL-5120
EL-5250
EL-9000
EL-9300
EL-9400
EL-9600
EL-9900
IQ-7100
IQ-7520
IQ-7720
IQ-8500
IQ-8900
IQ-9200
PC-1100
PC-1150
PC-1201
PC-1211
PC-1212
PC-1245
PC-1246
PC-1247
PC-1248
PC-1250A
PC-1251
PC-1262

PC-1270
PC-1280
PC-1285
PC-1300S
PC-1350
PC-1360
PC-1365
PC-1401
PC-1402
PC-1403
PC-1430
PC-1440
PC-1445
PC-1450
PC-1460
PC-1475
PC-1490UII
PC-1500A
PC-1600
PC-E220
PC-E500S
PC-G801
PC-G850
PC-G850S
PC-G850V
PC-G850VS

HEWLETT - PACKARD

HP-11C
HP-12C
HP-12C Platinum
HP-12C ARM
HP-15C
HP-15C LE
HP-16C
HP-17BII
HP-17BII+
HP-19BII
HP-20S
HP-28C
HP-28S
HP-29C
HP-30B
HP-32S
HP-32SII
HP-33C
HP-33S
HP-34C
HP-35S
HP-38G
HP-39GII
HP-40GS
HP-41C
HP-41CV
HP-41CX
HP-41CY
HP-41CL
HP-42S
HP-48GX
HP-48S
HP-48SX
HP-49G
HP-49G+
HP-50G
HP-67
HP-71B
HP-75C
HP-75D
HP-200LX
HP-Prime
WP-34S
DM-12CC
DM-15CC

TEXAS INSTRUMENTS

CC-40

SR-52
SR-56
TI-57
TI-57 II
TI-58C
TI-59
TI-62
TI-65
TI-66
TI-67
TI-73
TI-74 BASICALC
TI-80
TI-81
TI-82
TI-83+
TI-83+ SE
TI-84+
TI-84+ Pocket SE
TI-85
TI-86
TI-88 (Prototype)
TI-89
TI-89 Titanium
TI-92+
TI-95 PROCALC
TI-Nspire
TI-Nspire CAS
TI-Nspire CX CAS

CITIZEN

SRP-60
SRP-175
SRP-320G
SRP-325G
SRP-350
SRP-400G

ELEKTRONIKA

MK-52
MK-61
MK-85
MK-90
MK-152
MK-161

PSION

Organiser II XP
Series 3a
Series 3mx
Series 5mx

CANON

F-700
F-800P
X-07

COMMODORE

C64
P-50
PR-100

TRULY

SC-105
SC-185
TG-202

AURORA

AC-692
FN-1000

DURABRAND

828 (SC-828)
828 (SC-1376)

PRIVILEG

PR-56D-NC
SR-12PR

SINCLAIR

Cambridge Programmable
Enterprise Programmable

ASSISTANT

AC-3270

COMPUCESSORY

CCS28956

CORNER OFFICE

ATC-139

NEC

PC-2001

PARS HESAB

PX-5600PV

PROFF

SC-3610

SANCO

TPC-8300

SEIKO

UC-2200

TOSHIBA

IHC-8000 Pasopia Mini

VICTOR

V12

Basic

```

10  CLEAR:DEFINT A-Z          (DEFINT used if possible)
20  R=8                        (R>=1)
30  REM DIM A(R)              (DIM used if necessary)
40  IF X=R THEN 180           (140 for all solutions)
50  X=X+1
60  A(X)=R
70  S=S+1
80  Y=X
90  Y=Y-1
100 IF Y=0 THEN 40
110 T=A(X)-A(Y)
120 IF T=0 THEN 140
130 IF X-Y<>ABS(T) THEN 90
140 A(X)=A(X)-1
150 IF A(X)<>0 THEN 70         (<>0 omitted if possible)
160 X=X-1
170 IF X<>0 THEN 140          (<>0 omitted if possible)
180 PRINT S

```

```

>RUN
>876 (Nodes evaluated)
>_

```

(15720 nodes for all solutions)

Pascal

```

-----
procedure qbench;
var
  a:array[0..8] of integer;
  r,s,t,x,y:integer;
begin
  r:=8;
  s:=0;
  x:=0;
  repeat
    inc(x);
    a[x]:=r;
    repeat
      inc(s);
      y:=x;
      while y>1 do begin
        dec(y);
        t:=a[x]-a[y];
        if (t=0) or (x-y=abs(t)) then begin
          y:=0;
          dec(a[x]);
          while a[x]=0 do begin
            dec(x);
            dec(a[x]);
          end;
        end;
      end;
    until y=1;
    until x=r;
    writeln(s);
  end;

```

(Used as pattern for languages faster with structured code or missing GOTO)

C / Unstructured

```

main()
{

```

```
    int x,y,r,s,t,n,a[9];

    n=1000;
l5:  r=8;
    s=0;
    x=0;
l0:  if (x==r) goto l4;
    a[++x]=r;
l1:  ++s;
    y=x;
l2:  if (!--y) goto l0;
    if (!(t=a[x]-a[y])) goto l3;
    if (x-y!=abs(t)) goto l2;
l3:  if (--a[x]) goto l1;
    if (--x) goto l3;
l4:  if (--n) goto l5;
    printf("%d",s);
}
```

=====

C / Structured

```
main()
{
    int x,y,r,s,t,n,a[9];

    for(n=1000;n>0;--n){
        r=8;
        s=0;
        x=0;
        do{
            a[++x]=r;
            do{
                ++s;
                y=x;
                while(y>1)
                    if (!(t=a[x]-a[--y]) || x-y==abs(t)){
                        y=0;
                        while(!--a[x])
                            --x;
                    }
            } while(y!=1);
        } while(x!=r);
    }
    printf("%d",s);
}
```

(Used if faster than unstructured version)

=====

C / RegVars

```
main()
{
    register int x, y, r, s, t, n, Ax;
    int a[9];

    for (n = 1000000; n > 0; --n) {
        r = 8;
        s = 0;
        x = 0;
        Ax = 0;
        do {
            a[x] = Ax;
            ++x;
            Ax = r;
            do {
                ++s;
                y = x;
            }
        }
    }
}
```

```

        while (y > 1) {
            --y;
            if (!(t = Ax - a[y]) || x - y == abs(t)) {
                y = 0;
                while (!--Ax) {
                    --x;
                    Ax = a[x];
                }
            }
        }
    } while (y != 1);
} while (x != r);
}
return s;
}
```

(Optimized HP-50G version)

=====

Lua

```

local a,r,s,t,x,y,n
for n=1,100 do
    r=8
    s=0
    x=0
    a={}
    repeat
        x=x+1
        a[x]=r
        repeat
            s=s+1
            y=x
            while y>1 do
                y=y-1
                t=a[x]-a[y]
                if t<0 then
                    t=-t
                end
                if (t==0) or (x-y==t) then
                    y=0
                    a[x]=a[x]-1
                    while a[x]==0 do
                        x=x-1
                        a[x]=a[x]-1
                    end
                end
            end
        end
        until y==1
    until x==r
end
print(s)
```

=====

OPL

```

proc qbench:
global a%(8),r%,s%,t%,x%,y%,n%

    n%=100
l5:: r%=8
    s%=0
    x%=0
l0:: if x%=r%
    goto l4
    endif
    x%=x%+1
    a%(x%)=r%
l1:: s%=s%+1
```

```

      y%=x%
l2:: y%=y%-1
      if y%=0
          goto l0
      endif
      t%=a%(x%)-a%(y%)
      if t%=0
          goto l3
      endif
      if (x%-y%)<>iabs(t%)
          goto l2
      endif
l3:: a%(x%)=a%(x%)-1
      if a%(x%)<>0
          goto l1
      endif
      x%=x%-1
      if x%<>0
          goto l3
      endif
l4:: n%=n%-1
      if n%<>0
          goto l5
      endif
      print s%
      pause 0
endp
```

=====
MK-52
MK-61
MK-152
MK-161

- 00. 0D CX
- 01. 49 M9
- 02. 4C MC
- 03. 08 8
- 04. 4B MB
- 05. 69 RM9
- 06. 6B RMB
- 07. 11 -
- 08. 57 F X<>0
- 09. 52 52
- 10. 69 RM9
- 11. 01 1
- 12. 10 +
- 13. 49 M9
- 14. 6B RMB
- 15. B9 K M9
- 16. 6C RMC
- 17. 01 1
- 18. 10 +
- 19. 4C MC
- 20. 69 RM9
- 21. 4A MA
- 22. 6A RMA
- 23. 01 1
- 24. 11 -
- 25. 4A MA
- 26. 57 F X<>0
- 27. 05 05
- 28. D9 K RM9
- 29. DA K RMA
- 30. 11 -
- 31. 57 F X<>0
- 32. 40 40
- 33. 31 K ABS
- 34. 6A RMA
- 35. 10 +
- 36. 69 RM9
- 37. 11 -
- 38. 5E F X=0
- 39. 22 22
- 40. D9 K RM9

41. 01 1
42. 11 -
43. B9 K M9
44. 5E F X=0
45. 16 16
46. 69 RM9
47. 01 1
48. 11 -
49. 49 M9
50. 5E F X=0
51. 40 40
52. 6C RMC
53. 50 R/S

=====

ATC-139
828 (SC-1376)

```

      Mcl
      8->R
Lbl 0  X=R=>Goto 4
      Isz X
      R->A[X]
Lbl 1  Isz S
      X->Y
Lbl 2  Dsz Y Deg
      Y=0=>Goto 0
      A[X]-A[Y]->T
      T=0=>Goto 3
      X-Y<>Abs T=>Goto 2
Lbl 3  A[X]-1->A[X]
      Ans<>0=>Goto 1
      Dsz X Goto 3
Lbl 4  S
```

=====

SRP-320G
828 (SC-828)

```

      Mcl
      8->R
Lbl 0  X=R=>Goto 4
      Isz X
      R->A[X]
Lbl 1  Isz S
      X->Y
Lbl 2  Y-1->Y
      Y=0=>Goto 0
      A[X]-A[Y]->T
      T=0=>Goto 3
      X-Y<>Abs T=>Goto 2
Lbl 3  A[X]-1->A[X]
      A[X]<>0=>Goto 1
      Dsz X Goto 3
Lbl 4  S
```

=====

SRP-400G

```

R=8;
Seq(0,t,1,R,1)->List1;
S=0;
D=0;
For(X=0;X!=R;D++)
  X++;
```

```
R->List1[X];
For(Y=0;Y!=1;D++)
  S++;
  Y=X;
  For(D=0;Y>1;D++)
  { Y--;
    List1[X]->P; List1[Y]->Q; T=P-Q;
    If (T(X-Y-Abs(T))==0)
    { Y=0;
      List1[X]->P; P--; P->List1[X];
      For(D=0;P==0;D++)
      { X--;
        List1[X]->P; P--; P->List1[X];
      }
    }
  }
}
```

(GOTO not usable due to an interpreter bug)

EL-5030
EL-9000

```
8=>R
DIM A[R]
1=>S
1=>X
R=>A[X]
\->
  X+1=>X
  R=>A[X]
  \->
    S+1=>S
    X=>Y
    \->
      Y-1=>Y
      A[X]-A[Y]=>T
      T*(X-Y-ABS T)<>0=N=>[
        0=>Y
        A[X]-1=>A[X]]
      \->
        A[X]<>0=N=>[
          X-1=>X
          A[X]-1=>A[X]
          <- /]
      Y>1=Y=>[<- /]
      X>1=Y=>[Y-1=>Y]
      Y>=0=N=>[<- /]
      X<>R=Y=>[<- /]
S
```

(Adapted version due to several restrictions)

EL-9300

```
R=8
dim A[R]
S=0
X=0
Label 0 If X=R Goto 4
X=X+1
A[X]=R
Label 1 S=S+1
Y=X
```

```
Label 2  Y=Y-1
        If Y=0 Goto 0
        T=A[X]-A[Y]
        If T=0 Goto 3
        If X-Y<>abs T Goto 2
Label 3  A[X]=A[X]-1
        If A[X]<>0 Goto 1
        X=X-1
        If X Goto 3
Label 4  Print S
```

=====

EL-9400
EL-9600
EL-9900

```
      8=>R
      R=>dim(L1)
      0=>S
      0=>X
Label 0  If X=R Goto 4
      X+1=>X
      R=>L1(X)
Label 1  S+1=>S
      X=>Y
Label 2  Y-1=>Y
        If Y=0 Goto 0
        L1(X)-L1(Y)=>T
        If T=0 Goto 3
        If X-Y<>abs(T) Goto 2
Label 3  L1(X)-1=>L1(X)
        If L1(X) Goto 1
        X-1=>X
        If X Goto 3
Label 4  Print S
```

=====

PC-1300S

```
00; 8=>R,0=>S
01; DO 1,R
02; R=>XA
03; S+1=>S,A=>Y
04; Y-1=>Y,IF Y=0,GT0 9
05; IF XA-XY=>T,IF A-Y<>ABS T,GT0 4
06; IF XA-1=>XA,GT0 3
07; IF A-1=>A,GT0 6
08; GT0 10
09; CNT
10; S,END
```

=====

FX-201P

```
MAC
ST#0: I = I + K1:
      IM = K9:
      IF I = IM: 2: 5:
ST#1: I = I - K1:
ST#2: IM = IM - K1:
      9 = K0:
      IF 9 = IM: 3: 1:
ST#3: 9 = 9 + K1:
      IF 9 = I: 4: 0:
ST#4: 0 = IM:
```



```
I = I - 9:
0 = 0 - IM x =:
I = I + 9:
0 = 9 x 9 - 0 x 0:
IF 0 = K0: 3: 2: 3:
```

(Adapted version due to insufficient memory)

FX-502P

```
AC MAC
8 MinF
LBL0 MR0 x=F GOT04
ISZ
MRF IND Min0
LBL1 1 M+,F
MR0 Min9
LBL2 1 M-9
MR9 x=0 GOT00
IND MR0 - IND MR9 =
x=0 GOT03
ABS + MR9 - MR0 =
x=0 GOT03
GOT02
LBL3 IND DSZ GOT01
DSZ GOT03
LBL4 MR.F
```

FX-602P
FX-603P

```
AC MAC
8 MinF
LBL0 MR00 x=F GOT04
ISZ
MRF IND Min00
LBL1 1 M+1F
MR00 Min09
LBL2 1 M-09
MR09 x=0 GOT00
IND MR00 - IND MR09 =
x=0 GOT03
ABS + MR09 - MR00 =
x=0 GOT03
GOT02
LBL3 IND DSZ GOT01
DSZ GOT03
LBL4 MR1F
```

- FX-4000P
- FX-6000G
- FX-6300G
- FX-6500G
- FX-7000G
- FX-7000GA
- FX-7000GB
- FX-7200G
- FX-7300G
- FX-7500G
- FX-7700GB
- FX-7700GH
- FX-8000G
- FX-8500G

FX-8700G
FX-9700GE
FX-9700GH
CFX-9800G
OH-7000G
TG-202

```

      Mcl
      8->R
Lbl 0  X=R=>Goto 4
      Isz X
      R->A[X]
Lbl 1  Isz S
      X->Y
Lbl 2  Dsz Y Deg
      Y=0=>Goto 0
      A[X]-A[Y]->T
      T=0=>Goto 3
      X-Y<>Abs T=>Goto 2
Lbl 3  Dsz A[X] Goto 1
      Dsz X Goto 3
Lbl 4  S
```

FC-1000

```

      Mcl
      8->R
Lbl 0  X=R=>Goto 4
      Isz X
      R->CF(X)
Lbl 1  Isz S
      X->Y
Lbl 2  Dsz Y Norm
      Y=0=>Goto 0
      CF(X)-CF(Y)->T
      T=0=>Goto 3
      X-Y<>Abs T=>Goto 2
Lbl 3  CF(X)-1->CF(X)
      CF(X)<>0=>Goto 1
      Dsz X Goto 3
Lbl 4  S
```

FX-4500P
FX-4500PA
PX-5600PV
SF-9350 / SF-R10 / Scientific ROM Card ES-650

```

      Mcl
      Fixm
      R=8
Lbl 0  X=R=>Goto 4_
      X=X+1
      A[X]=R
Lbl 1  S=S+1
      Y=X
Lbl 2  Y=Y-1
      Y=0=>Goto 0_
      T=A[X]-A[Y]
      T=0=>Goto 3_
      X-Y<>Abs T=>Goto 2_
Lbl 3  A[X]=A[X]-1
      Ans<>0=>Goto 1_
      X=X-1
      X<>0=>Goto 3_
Lbl 4  S
```

=====

FX-4800P
FX-4850P

```

      Mcl
      Fixm
      R=8
Lbl 0  X=R=>Goto 4_
      Isz X
      A[X]=R
Lbl 1  Isz S
      Y=X
Lbl 2  Dsz Y Deg
      Y=0=>Goto 0_
      T=A[X]-A[Y]
      T=0=>Goto 3_
      X-Y<>Abs T=>Goto 2_
Lbl 3  Dsz A[X] Goto 1
      Dsz X Goto 3
Lbl 4  S
=====
```

FX-5800P

```

      ClrMemory
      8->R
      R->DimZ
Lbl 0  X=R=>Goto 4
      Isz X
      R->Z[X]
Lbl 1  Isz S
      X->Y
Lbl 2  Dsz Y Deg
      Y=0=>Goto 0
      Z[X]-Z[Y]->T
      T=0=>Goto 3
      X-Y<>Abs(T)=>Goto 2
Lbl 3  Z[X]-1->Z[X]
      Ans=>Goto 1
      Dsz X Goto 3
Lbl 4  S
=====
```

FX-7400G+
FX-7400G+ GY
FX-7450G
FX-9750G
FX-9750G+
RM-9850G
CFX-9850GB+
CFX-9950GB+
CFX-9970G

```

      0->A~Z
      8->R
      Seq(0,R,1,R,1)->List 1
Lbl 0  X=R=>Goto 4
      Isz X
      R->List 1[X]
Lbl 1  Isz S
      X->Y
Lbl 2  Dsz Y Deg
      Y=0=>Goto 0
      List 1[X]-List 1[Y]->T
      T=0=>Goto 3
=====
```

```

      X-Y<>Abs T=>Goto 2
Lbl 3 List 1[X]-1->List 1[X]
      List 1[X]=>Goto 1
      Dsz X Goto 3
Lbl 4 S
```

FX-9750G II
FX-9860G SD
FX-9860G Slim
FX-9860G II
FX-CG20

```

0->A~Z
8->R
{R,1}->Dim Mat A
0
Do
  Isz X
  R->Mat A[X,1]
  Do
    Isz S
    X->Y
    While Y>1
      Dsz Y
      Mat A[X,1]-Mat A[Y,1]->T
      If T=0 Or X-Y=Abs T
        Then 0->Y
        Mat A[X,1]-1->Mat A[X,1]
        While Mat A[X,1]=0
          Dsz X
          Mat A[X,1]-1->Mat A[X,1]
        WhileEnd
      IfEnd
    WhileEnd
  LpWhile Y<>1
LpWhile X<>R
S
```

AFX-2.0+

```

0->A~Z
8->R
R->Dim List 1
0
Isz X
R->List 1[X]
Do
  Isz S
  X->Y
  While Y>1
    Dsz Y
    List 1[X]-List 1[Y]->T
    If T=0 Or X-Y=Abs T
      Then 0->Y
      List 1[X]-1->List 1[X]
      While List 1[X]=0
        Dsz X
        List 1[X]-1->List 1[X]
      WhileEnd
    IfEnd
  WhileEnd
LpWhile Y<>1
LpWhile X<>R
S
```

ClassPad 300+
ClassPad 330
FX-CP400

```
{}=>a
8=>r
0=>s
0=>x
Lbl a  If x=r Then
      Goto e
      IfEnd
      x+1=>x
      r=>a[x]
Lbl b  s+1=>s
      x=>y
Lbl c  y-1=>y
      If y=0 Then
        Goto a
      IfEnd
      a[x]-a[y]=>t
      If t=0 Then
        Goto d
      IfEnd
      If x-y<>abs(t) Then
        Goto c
      IfEnd
Lbl d  a[x]-1=>a[x]
      If a[x] Then
        Goto b
      IfEnd
      x-1=>x
      If x Then
        Goto d
      IfEnd
Lbl e  Print s
```

SR-52

```
CLR CMS
8 STO 11
LBL A  RCL 00 - RCL 11 =
      IFZRO E
      1 SUM 00
      RCL 11 IND STO 00
LBL B  1 SUM 10
      RCL 00 STO 09
LBL C  1 INV SUM 09
      RCL 09 IFZRO A
      IND RCL 00 - IND RCL 09 =
      IFZRO D
      IFPOS +/- +/- LBL +/-
      + RCL 09 - RCL 00 =
      INV IFZRO C
LBL D  1 IND INV SUM 00
      IND RCL 00 INV IFZRO B
      DSZ D
LBL E  RCL 10
      HLT
```

TI-58C
TI-59
TI-66

```
CLR CMS
8 STO 11
```

```
LBL A  RCL 00 X<->T
      RCL 11 X=T E
      CLR X<->T
      OP 20
      RCL 11 STO IND 00
LBL B  1 SUM 10
      RCL 00 STO 09
LBL C  INV DSZ 09 A
      RCL IND 00 - RCL IND 09 =
      X=T D
      ABS + RCL 09 - RCL 00 =
      INV X=T C
LBL D  DSZ IND 00 B
      DSZ 00 D
LBL E  RCL 10
      R/S
```

TI-88

```
CE CLR CMs
8 Sto R
Lbl A  Rcl R If= X Gto E
      1 St+ X
      Rcl R Sto Ind x
Lbl B  1 St+ S
      Rcl X Sto Y
Lbl C  Inv Dsz Y Gto A
      Rcl Ind x - Rcl Ind y =
      If= Z Gto D
      OP26 + Y - X =
      Inv If= Z Gto C
Lbl D  Dsz Ind x Gto B
      Dsz X Gto D
Lbl E  Rcl S
      R/S
```

(Never released prototype)

TI-95 PROCALC

```
CLR CMS
8 STO R
LBL 00  RCL R IF= X GTL 04
      INC X
      STO IND X
LBL 01  INC S
      RCL X STO Y
LBL 02  INV DSZ Y GTL 00
      RCL IND X - RCL IND Y =
      IF= Z GTL 03
      ABS + RCL Y - RCL X =
      INV IF= Z GTL 02
LBL 03  DSZ IND X GTL 01
      DSZ X GTL 03
LBL 04  RCL S
      HLT
```

(ASM and UNF used for faster execution)

TI-74 BASICALC / Learn Pascal ROM Module

```
100 program qbench;
110 procedure main;
120   label 0,1,2,3,4;
130   var a:array[0..8] of integer;
140       n,r,s,t,x,y:integer;
150 begin
160   r:=8;
170   s:=0;
180   x:=0;
190 0:if x=r then goto 4;
200   x:=succ(x);
210   a[x]:=r;
220 1:s:=succ(s);
230   y:=x;
240 2:y:=pred(y);
250   if y=0 then goto 0;
260   t:=a[x]-a[y];
270   if t=0 then goto 3;
280   if x-y<>abs(t) then goto 2;
290 3:a[x]:=pred(a[x]);
300   if a[x]<>0 then goto 1;
310   x:=pred(x);
320   if x<>0 then goto 3;
330 4:write(s);
340 end;
350 begin
360   main;
370 end.
```

TI-80

```
8->R
R->DIM L1
0->S
0->X
LBL A  IF X=R GOTO E
X+1->X
R->L1(X)
LBL B  S+1->S
X->Y
LBL C  Y-1->Y
IF Y=0 GOTO A
L1(X)-L1(Y)->T
IF T=0 GOTO D
IF X-Y<>ABS T GOTO C
LBL D  L1(X)-1->L1(X)
IF ANS GOTO B
X-1->X
IF X GOTO D
LBL E  DISP S
```

TI-81

```
8->R
0->S
0->X
Lbl A  If X=R Goto E
X+1->X
R->{X}(X)
Lbl B  S+1->S
X->Y
Lbl C  Y-1->Y
If Y=0 Goto A
{X}(X)-{X}(Y)->T
If T=0 Goto D
If X-Y<>abs T Goto C
Lbl D  {X}(X)-1->{X}(X)
```

```
      If Ans Goto B
      DS<(X,1) Goto D
Lbl E  Disp S
```

=====

```
TI-73
TI-82
TI-83+
TI-83+ SE
TI-84+
TI-84+ Pocket SE
TI-Nspire / TI-84+ Keypad
-----
```

```
      8->R
      R->dim(L1)
      0->S
      0->X
Lbl A  If X=R Goto E
      X+1->X
      R->L1(X)
Lbl B  S+1->S
      X->Y
Lbl C  Y-1->Y
      If Y=0 Goto A
      L1(X)-L1(Y)->T
      If T=0 Goto D
      If X-Y<>abs(T) Goto C
Lbl D  L1(X)-1->L1(X)
      If Ans Goto B
      DS<(X,1) Goto D
Lbl E  Disp S
```

=====

```
TI-85
TI-86
-----
```

```
      8->R
      R->dimL A
      0->S
      0->X
Lbl A  If X==R Goto E
      X+1->X
      R->A(X)
Lbl B  S+1->S
      X->Y
Lbl C  Y-1->Y
      If Y==0 Goto A
      A(X)-A(Y)->T
      If T==0 Goto D
      If X-Y<>abs T Goto C
Lbl D  A(X)-1->A(X)
      If Ans Goto B
      DS<(X,1) Goto D
Lbl E  Disp S
```

=====

```
TI-89
TI-89 Titanium
TI-92+
TI-Nspire
TI-Nspire CAS
-----
```

```
Local a,r,s,t,x,y
8->r
newList(r)->a
```



```
0->s
0->x
Loop
  x+1->x
  r->a[x]
  Loop
    s+1->s
    x->y
    While y>1
      y-1->y
      a[x]-a[y]->t
      If t=0 or x-y=abs(t) Then
        0->y
        a[x]-1->a[x]
        While a[x]=0
          x-1->x
          a[x]-1->a[x]
        EndWhile
      EndIf
    EndWhile
  EndLoop
  If x=r
    Exit
  EndLoop
Disp s
```

(Auto or Exact Mode)

=====

HP-67

```
LBL A  CLEAR REG
      8 STO A
LBL 0   RCL 0 RCL A
      x=y? GT0 4
      1 STO+ 0
      RCL 0 STO I
      RCL A STO(i)
LBL 1   RCL B 1 + STO B
      RCL 0 STO 9
LBL 2   1 STO- 9
      RCL 9 x=0? GT0 0
      RCL 0 STO I RCL(i)
      RCL 9 STO I
      RV RCL(i) -
      x=0? GT0 3
      ABS RCL 0 RCL 9 -
      x<>y? GT0 2
LBL 3   RCL 0 STO I
      1 STO-(i)
      RCL(i) x<>0? GT0 1
      1 STO- 0
      RCL 0 x<>0? GT0 3
LBL 4   RCL B
      RTN
```

=====

HP-11C
HP-29C
HP-34C

```
LBL A  CLEAR REG
      8 STO .0
LBL 0   RCL 0 RCL .0
      x=y? GT0 4
      1 STO+ 0
      RCL 0 STO I
```

```

RCL .0 STO(i)
LBL 1 RCL .1 1 + STO .1
      RCL 0 STO 9
LBL 2 1 STO- 9
      RCL 9 x=0? GT0 0
      RCL 0 STO I RCL(i)
      RCL 9 STO I
      RV RCL(i) -
      x=0? GT0 3
      ABS RCL 0 RCL 9 -
      x<>y? GT0 2
LBL 3 RCL 0 STO I
      1 STO-(i)
      RCL(i) x<>0? GT0 1
      1 STO- 0
      RCL 0 x<>0? GT0 3
LBL 4 RCL .1
      RTN
```

HP-15C
HP-15C LE
DM-15CC

```

LBL A CLEAR REG
      8 STO .0
LBL 0 RCL 0 RCL .0
      TEST 5 GT0 4
      1 STO+ 0
      RCL 0 STO I
      RCL .0 STO(i)
LBL 1 1 STO+ .1
      RCL 0 STO 9
LBL 2 1 STO- 9
      RCL 9 x=0? GT0 0
      RCL 0 STO I RCL(i)
      RCL 9 STO I
      RV RCL(i) -
      x=0? GT0 3
      ABS RCL 0 RCL 9 -
      TEST 6 GT0 2
LBL 3 RCL 0 STO I
      1 STO-(i)
      RCL(i) TEST 0 GT0 1
      1 STO- 0
      RCL 0 TEST 0 GT0 3
LBL 4 RCL .1
      RTN
```

HP-16C

```

LBL A FLOAT 0
      CLEAR REG
      8 STO .0
LBL 0 RCL 0 RCL .0
      x=y? GT0 4
      RCL 0 1 + STO 0
      STO I RCL .0 STO(i)
LBL 1 RCL .1 1 + STO .1
      RCL 0 STO 9
LBL 2 RCL 9 1 - STO 9
      x=0? GT0 0
      RCL 0 STO I RCL(i)
      RCL 9 STO I
      RV RCL(i) -
      x=0? GT0 3
      ABS RCL 0 RCL 9 -
      x<>y? GT0 2
LBL 3 RCL 0 STO I
```

```

RCL(i) 1 - STO(i)
x<>0? GT0 1
RCL 0 1 - STO 0
x<>0? GT0 3
LBL 4 RCL .1
RTN
```

=====

HP-32S
HP-32SII
HP-33S

```

LBL Q CLVARS
8 STO R
LBL A RCL X RCL R
x=y? GT0 E
1 STO+ X
RCL X STO i
RCL R STO(i)
LBL B 1 STO+ S
RCL X STO Y
LBL C 1 STO- Y
RCL Y x=0? GT0 A
RCL X STO i RCL(i)
RCL Y STO i
RV RCL(i) -
x=0? GT0 D
ABS RCL X RCL Y -
x<>y? GT0 C
LBL D RCL X STO i
1 STO-(i)
RCL(i) x<>0? GT0 B
1 STO- X
RCL X x<>0? GT0 D
LBL E RCL S
RTN
```

=====

HP-41C
HP-41CV
HP-41CX
HP-41CY
HP-41CL
HP-42S
HP-71B / 41-Translator ROM Module HP-82490A

```

LBL A CLRG
8 STO 11
LBL 00 RCL 00 RCL 11
X=Y? GT0 04
ISG 00 DEG
STO IND 00
LBL 01 ISG 10 DEG
RCL 00 STO 09
LBL 02 DSE 09 DEG
RCL 09 X=0? GT0 00
RCL IND 00 RCL IND 09 -
X=0? GT0 03
ABS RCL 00 RCL 09 -
X<>Y? GT0 02
LBL 03 DSE IND 00 GT0 01
DSE 00 GT0 03
LBL 04 RCL 10
RTN
```

=====

HP-12C

HP-12C Platinum
HP-12C ARM
DM-12CC
V12
FN-1000
CCS28956

```
01  Clear Sigma
02  8
03  STO 0
04  RCL 1
05  RCL 0
06  x<=y
07  GTO 58
08  1
09  STO+1
10  RCL 1
11  STO n
12  RCL 0
13  STO Nj
14  1
15  STO+2
16  RCL 1
17  STO 3
18  1
19  STO -3
20  RCL 3
21  x=0
22  GTO 04
23  STO n
24  RCL Nj
25  RCL 1
26  STO n
27  RV
28  RCL Nj
29  -
30  x=0
31  GTO 43
32  0
33  x<->y
34  x<=y
35  CHS
36  RCL 3
37  +
38  RCL 1
39  -
40  x=0
41  GTO 43
42  GTO 18
43  RCL 1
44  STO n
45  RCL Nj
46  1
47  -
48  STO Nj
49  x=0
50  GTO 52
51  GTO 14
52  1
53  STO -1
54  RCL 1
55  x=0
56  GTO 58
57  GTO 43
58  RCL 2
59  GTO 00
```

=====
HP - 30B

```
01  8
02  STO 1
03  0
```

```
04  STO 2
05  STO 3
06  Lbl 00
07  RCL 2
08  RCL 1
09  ?=
10  GT 04
11  ISG 2
12  Off
13  RCL 2
14  STO 0
15  RCL 1
16  STO Data
17  Lbl 01
18  ISG 3
19  Off
20  RCL 2
21  STO 4
22  Lbl 02
23  DSE 4
24  RCL 4
25  RCL 4
26  GF 00
27  STO 0
28  RCL Data
29  RCL 2
30  STO 0
31  ( Rv
32  RCL Data
33  -
34  Input
35  GF 03
36  Math
37  Down
38  Down
39  Down
40  =
41  RCL 2
42  RCL 4
43  -
44  -
45  GT 02
46  Lbl 03
47  RCL 2
48  STO 0
49  RCL Data
50  1
51  -
52  STO Data
53  GT 01
54  DSE 2
55  Gto 03
56  Lbl 04
57  RCL 3
58  Stop
```

WP-34S

```
001: LBL B
002: CLREG
003: 8
004: STO 11
005: RCL 11
006: x=? 00
007: SKIP 22
008: INC 00
009: STO ->00
010: INC 10
011: RCL 00
012: STO 09
013: DEC 09
014: RCL 09
015: x=0?
016: BACK 11
```

```
017: RCL ->00
018: RCL - ->09
019: x=0?
020: SKIP 05
021: ABS
022: RCL 00
023: RCL - 09
024: x<>? Y
025: BACK 12
026: DSZ ->00
027: BACK 17
028: DSZ 00
029: BACK 03
030: RCL 10
031: RTN
```

HP -35S

```
Q001 LBL Q
Q002 CLVARS
Q003 8
Q004 STO R
Q005 1
Q006 STO K
Q007 RCL I
Q008 RCL R
Q009 x=y?
Q010 GTO Q039
Q011 RCL K
Q012 STO+ I
Q013 RCL R
Q014 STO(I)
Q015 RCL K
Q016 STO+ S
Q017 RCL I
Q018 STO J
Q019 RCL K
Q020 STO- J
Q021 RCL J
Q022 x=0?
Q023 GTO Q007
Q024 RCL(I)
Q025 RCL(J)
Q026 -
Q027 x=0?
Q028 GTO Q035
Q029 ABS
Q030 RCL I
Q031 RCL J
Q032 -
Q033 x<>y?
Q034 GTO Q019
Q035 DSE(I)
Q036 GTO Q015
Q037 DSE I
Q038 GTO Q035
Q039 RCL S
Q040 RTN
```

HP -17BII
HP -17BII+
HP -19BII

```
QUEENS:
Z=A+B+C+D+E+F+G+H+
Sigma(I:1:8:1:
Sigma(J:1:8:1:
IF(I<>J AND SQ(I-J)<>1:
```

```
Sigma(K:1:8:1:
IF(I<>K AND SQ(I-K)<>4
AND J<>K AND SQ(J-K)<>1:
Sigma(L:1:8:1:
IF(I<>L AND SQ(I-L)<>9
AND J<>L AND SQ(J-L)<>4
AND K<>L AND SQ(K-L)<>1:
Sigma(M:1:8:1:
IF(I<>M AND SQ(I-M)<>16
AND J<>M AND SQ(J-M)<>9
AND K<>M AND SQ(K-M)<>4
AND L<>M AND SQ(L-M)<>1:
Sigma(N:1:8:1:
IF(I<>N AND SQ(I-N)<>25
AND J<>N AND SQ(J-N)<>16
AND K<>N AND SQ(K-N)<>9
AND L<>N AND SQ(L-N)<>4
AND M<>N AND SQ(M-N)<>1:
Sigma(O:1:8:1:
IF(I<>O AND SQ(I-O)<>36
AND J<>O AND SQ(J-O)<>25
AND K<>O AND SQ(K-O)<>16
AND L<>O AND SQ(L-O)<>9
AND M<>O AND SQ(M-O)<>4
AND N<>O AND SQ(N-O)<>1:
Sigma(P:1:8:1:
IF(I<>P AND SQ(I-P)<>49
AND J<>P AND SQ(J-P)<>36
AND K<>P AND SQ(K-P)<>25
AND L<>P AND SQ(L-P)<>16
AND M<>P AND SQ(M-P)<>9
AND N<>P AND SQ(N-P)<>4
AND O<>P AND SQ(O-P)<>1:
L(A:I)xL(B:J)xL(C:K)xL(D:L)xL(E:M)xL(F:N)xL(G:O)xL(H:P)/0
:0)):0)):0)):0)):0)):0)):0)))
```

HP-38G
HP-40GS

```
8->R:
MAKELIST(0,X,1,R,1)->L1:
0->S:
0->X:
DO
  X+1->X:
  R->L1(X):
  DO
    S+1->S:
    X->Y:
    WHILE Y>1 REPEAT
      Y-1->Y:
      L1(X)-L1(Y)->T:
      IF T==0 OR X-Y==ABS(T) THEN
        0->Y:
        L1(X)-1->L1(X):
        WHILE L1(X)==0 REPEAT
          X-1->X:
          L1(X)-1->L1(X):
        END:
      END:
    END:
  UNTIL Y==1 END:
UNTIL X==R END:
MSGBOX S:
```

HP-39GII
HP-Prime

```
EXPORT NQUEENS()  
BEGIN  
  R:=8;  
  L1:=MAKELIST(0,X,1,R,1);  
  S:=0;  
  X:=0;  
  REPEAT  
    X:=X+1;  
    L1(X):=R;  
    REPEAT  
      S:=S+1;  
      Y:=X;  
      WHILE Y>1 DO  
        Y:=Y-1;  
        T:=L1(X)-L1(Y);  
        IF T==0 OR X-Y==ABS(T) THEN  
          Y:=0;  
          L1(X):=L1(X)-1;  
          WHILE L1(X)==0 DO  
            X:=X-1;  
            L1(X):=L1(X)-1;  
          END;  
        END;  
      END;  
    END;  
  UNTIL Y==1 END;  
  UNTIL X==R END;  
  S;  
END;
```

=====

HP-28C

```
<< 8 0 0 0 0 -> R S X Y A  
<< CLEAR  
1 R START 0 NEXT R ->LIST 'A' STO  
DO  
  X 1 + 'X' STO  
  A X R PUT 'A' STO  
DO  
  S 1 + 'S' STO  
  X 'Y' STO  
  WHILE Y 1 > REPEAT  
    Y 1 - 'Y' STO  
    A X GET A Y GET -  
    IF DUP 0 == SWAP ABS X Y - == OR THEN  
      0 'Y' STO  
      A X A X GET 1 - PUT 'A' STO  
      WHILE A X GET 0 == REPEAT  
        X 1 - 'X' STO  
        A X A X GET 1 - PUT 'A' STO  
      END  
    END  
  END  
  UNTIL Y 1 == END  
  UNTIL X R == END  
S  
>>  
>>
```

=====

HP-28S

```
<< 8 0 0 0 0 -> R S X Y A  
<< CLEAR # 11CAh SYSEVAL  
1 R START 0 NEXT R ->ARRY 'A' STO  
DO  
  X 1 + 'X' STO  
  A X R PUT 'A' STO  
DO  
  S 1 + 'S' STO
```



```

X 'Y' STO
WHILE Y 1 > REPEAT
Y 1 - 'Y' STO
A X GET A Y GET -
IF DUP 0 == SWAP ABS X Y - == OR THEN
0 'Y' STO
A X A X GET 1 - PUT 'A' STO
WHILE A X GET 0 == REPEAT
X 1 - 'X' STO
A X A X GET 1 - PUT 'A' STO
END
END
END
UNTIL Y 1 == END
UNTIL X R == END
# 11CAh SYSEVAL SWAP - B->R 8192 /
S
>>
>>
```

HP-48S
HP-48SX
HP-48GX
HP-49G
HP-49G+
HP-50G

```

<< 8. 0. 0. 0. { } -> R S X Y A
<< CLEAR TICKS
1. R START 0. NEXT R ->LIST 'A' STO
DO
'A' 'X' INCR R PUT
DO
'S' INCR DROP
X 'Y' STO
WHILE Y 1. > REPEAT
A X GET A 'Y' DECR GET -
IF DUP 0. == SWAP ABS X Y - == OR THEN
0. 'Y' STO
'A' X A X GET 1. - PUT
WHILE A X GET 0. == REPEAT
'A' 'X' DECR A X GET 1. - PUT
END
END
END
UNTIL Y 1. == END
UNTIL X R == END
TICKS SWAP - B->R 8192. /
S
>>
>>
```

HP-48GX / SysRPL

```

ASSEMBLE
NIBASC /HPHP48-R/
RPL
DEFINE A 1GETLAM
DEFINE A\<- 1PUTLAM
DEFINE Y 2GETLAM
DEFINE Y\<- 2PUTLAM
DEFINE X 3GETLAM
DEFINE X\<- 3PUTLAM
DEFINE S 4GETLAM
DEFINE S\<- 4PUTLAM
DEFINE R 5GETLAM
DEFINE R\<- 5PUTLAM
```

```
DOCOL CK0NOLASTWD
EIGHT
ZEROZEROZERO DUP
5PICK NDUPN {}N
NULLLAM FIVE NDUPN {}N BIND
CLKTICKS
BEGIN
( A[++X] = R )
R X #1+DUP X\<- A
PUTLIST A\<-
BEGIN
S #1+ S\<-
X Y\<-
BEGIN
Y #>1
WHILE
( | A[X] - A[--Y] | )
A X NTHCOMPDROP
A Y #1- DUP Y\<- NTHCOMPDROP
2DUP#< IT SWAP #-
DUP#0= SWAP
X Y #- #= OR
IT DOCOL
#0 Y\<-
BEGIN
( --A[X] )
A X NTHCOMPDROP
#1- DUP
X A PUTLIST A\<-
#0=
WHILE
X #1- X\<-
REPEAT
SEMI
REPEAT
Y #1=
UNTIL
X R #=
UNTIL
CLKTICKS SWAP bit- #>%
# 2000 UNCOERCE %/
A S
ABND
SEMI
```

=====

HP-71B / Forth / 41-Translator ROM Module HP-82490A

```
: NQUEENSF ;

FVARIABLE TIMER

: STARTTIMER CLOCK TIMER STO ;
: DISPLAYTIMER CLOCK TIMER RCL F- STD F. ." SEC" ;

8 CONSTANT RR
VARIABLE SS
VARIABLE XX
VARIABLE YY
CREATE AA RR 1 + ALLOT

: RCLAA @ 2 * AA + C@ ;
: STOA @ 2 * AA + C! ;

: NQCORE
0 SS !
0 XX !
BEGIN
1 XX +! RR XX STOA
BEGIN
1 SS +!
XX @ YY !
BEGIN YY @ 1 > WHILE
-1 YY +!
XX RCLAA YY RCLAA - DUP
```

```
0 = SWAP ABS XX @ YY @ - = OR IF
0 YY !
BEGIN XX RCLAA 1 - DUP XX ST0AA 0 = WHILE
-1 XX +!
REPEAT
THEN
REPEAT
YY @ 1 = UNTIL
RR XX @ = UNTIL
;

: NQUEENS
STARTTIMER
NOCORE
." S=" SS @ . ." IN "
DISPLAYTIMER CR
;
```

HP-41CX / HP-41CL / Mcode (NUT Assembly)

		.NAME	"CQ2"
*0078	0B2	#0B2	
*0079	011	#011	
*007A	003	#003	
007B	1A0	[CQ2] A=B=C=0	
007C	158	M=C	
007D	070	N=C	
007E	268	WRIT 9(Q)	
007F	228	WRIT 8(P)	
0080	1E8	WRIT 7(O)	
0081	1A8	WRIT 6(N)	
0082	168	WRIT 5(M)	
0083	128	WRIT 4(L)	
0084	0E8	WRIT 3(X)	
0085	0A8	WRIT 2(Y)	
0086	068	WRIT 1(Z)	
0087	2A0	SETDEC	
0088	198	[L40_2] C=M	
0089	0A6	A<>C S&X	
008A	130008	LDIS&X 008	
008C	366	?A#C S&X	
008D	093	JNC [SL180_2] +18 009F	
008E	198	[L50_2] C=M	
008F	226	C=C+1 S&X	
0090	158	M=C	
0091	198	[L60_2] C=M	
0092	270	RAMSLCT	
0093	130008	LDIS&X 008	
0095	2F0	WRITDATA	
0096	278	[L70_2] READ 9(Q)	
0097	22E	C=C+1 ALL	
0098	268	WRIT 9(Q)	
0099	198	[L80_2] C=M	
009A	070	N=C	
009B	0B0	[L90_2] C=N	
009C	266	C=C-1 S&X	
009D	070	N=C	
009E	013	JNC [L100_2] +2 00A0	
009F	12B	[SL180_2] JNC [L180_2] +37 00C4	
00A0	0B0	[L100_2] C=N	
00A1	2E6	?C#0 S&X	
00A2	333	JNC [L40_2] -26 0088	
00A3	198	[L110_2] C=M	
00A4	270	RAMSLCT	
00A5	038	READDATA	
00A6	0A6	A<>C S&X	
00A7	0B0	C=N	
00A8	270	RAMSLCT	
00A9	038	READDATA	
00AA	246	C=A-C S&X	
00AB	013	JNC (c6_2) +2 00AD	
00AC	286	C=0-C S&X	
00AD	2E6	(c6_2) ?C#0 S&X	
00AE	04B	JNC [L140_2] +9 00B7	

```
00AF 0E6      C<>B      S&X
00B0 198 [L130_2] C=M
00B1 0A6      A<>C      S&X
00B2 0B0      C=N
00B3 1C6      A=A - C   S&X
00B4 0E6      C<>B      S&X
00B5 366      ?A#C      S&X
00B6 32F      JC        [L90_2] -27 009B
00B7 198 [L140_2] C=M
00B8 270      RAMSLCT
00B9 038      READDATA
00BA 266      C=C - 1   S&X
00BB 2F0      WRITDATA
00BC 2E6 [L150_2] ?C#0   S&X
00BD 2CF      JC        [L70_2] -39 0096
00BE 198 [L160_2] C=M
00BF 266      C=C - 1   S&X
00C0 158      M=C
00C1 198 [L170_2] C=M
00C2 2E6      ?C#0   S&X
00C3 3A7      JC        [L140_2] -12 00B7
00C4 35C [L180_2] R=      12
00C5 278      READ      9(Q)
00C6 2FC (c8_2) RCR      13
00C7 2E2      ?C#0   @R
00C8 3F3      JNC      (c8_2) -2 00C6
00C9 0E8      WRIT      3(x)
00CA 3E0      RTN
```

HP-48GX / Saturn Assembly

```
:: CK0
* CLKTICKS
CODE
L10  GOSBVL =SAVPTR
    LAHEX 888888888
    R0=A
    C=0 W
    D1=C
    B=0 A
L40  P= 0
    A=R0
    ?A=B P
    GOYES L180
L50  B=B+1 P
L60  P= 0
    C=B P
    P=C 0
    C=A P
L70  CD1EX
    C=C+1 A
    CD1EX
L80  P= 0
    C=B P
    D=C P
L90  D=D - 1 P
L100 ?D=0 P
    GOYES L40
L110 P= 0
    C=D P
    GOSUB Ptst
    P= 0
    A=C P
    C=B P
    GOSUB Ptst
    P= 0
    ?A>=C P
    GOYES NoSwp
    ACEX P
NoSwp A=A - C P
L120 ?A=0 P
    GOYES L140
L130 C=B P
```

```
L140 C=C - D P
      ?A#C P
      GOYES L90
      P= 0
      C=B P
      P=C 0
      C=C - 1 P
L150 ?C#0 P
      GOYES L70
L160 P= 0
      B=B - 1 P
L170 ?B#0 P
      GOYES L140
L180 CSR W
      AD1EX
      P= 7
      ACEX WP
      RSTK=C
      GOSBVL =PUSHhxs
      GOSBVL =SAVPTR
      C=RSTK
      A=C A
Ptst GOVLNG =PUSH#AL00P
      P=C 0
      ?P# 1
      GOYES tP2
      CPEX 1
      C=P 0
      CPEX 1
      RTNCC
tP2 ?P# 2
      GOYES tP3
      CPEX 2
      C=P 0
      CPEX 2
      RTNCC
tP3 ?P# 3
      GOYES tP4
      CPEX 3
      C=P 0
      CPEX 3
      RTNCC
tP4 ?P# 4
      GOYES tP5
      CPEX 4
      C=P 0
      CPEX 4
      RTNCC
tP5 ?P# 5
      GOYES tP6
      CPEX 5
      C=P 0
      CPEX 5
      RTNCC
tP6 ?P# 6
      GOYES tP7
      CPEX 6
      C=P 0
      CPEX 6
      RTNCC
tP7 ?P# 7
      GOYES tP8
      CPEX 7
      C=P 0
      CPEX 7
      RTNCC
tP8 CPEX 8
      C=P 0
      CPEX 8
      RTNCC
ENDCODE
*
CLKTICKS
4ROLL
bit- #>% # 2000 UNCOERCE %/
SWAP UNCOERCE
;
```

=====

HP-50G / ARM Assembly

```
#define _ASM asm volatile

int benchmark_asm() __attribute__((naked));
int benchmark_asm()
{
    _ASM("STMDB SP!, { R1-R8 }");
    _ASM("MOV R3, #8");
    _ASM("SUB SP, SP, R3, LSL #2");
    _ASM("SUB SP, SP, #8");
    _ASM("MOV R4, #102400");
    _ASM("LOOPSTART:");
    _ASM("MOV R1, #0");
    _ASM("MOV R2, #0");
    _ASM("MOV R0, #0");
    _ASM("MOV R6, #0");
    _ASM("LINE40:");
    _ASM("CMP R1, R3");
    _ASM("BEQ LINE180");
    _ASM("STR R6, [SP, R1, LSL #2]");
    _ASM("ADD R1, R1, #1");
    _ASM("MOV R6, R3");
    _ASM("LINE70:");
    _ASM("ADD R0, R0, #1");
    _ASM("MOV R2, R1");
    _ASM("LINE90:");
    _ASM("SUBS R2, R2, #1");
    _ASM("BEQ LINE40");
    _ASM("LDR R7, [SP, R2, LSL #2]");
    _ASM("SUBS R7, R6, R7");
    _ASM("BEQ LINE140");
    _ASM("RSBMI R7, R7, #0");
    _ASM("SUB R8, R1, R2");
    _ASM("CMP R7, R8");
    _ASM("BNE LINE90");
    _ASM("LINE140:");
    _ASM("SUBS R6, R6, #1");
    _ASM("BNE LINE70");
    _ASM("SUBS R1, R1, #1");
    _ASM("LDR R6, [SP, R1, LSL #2]");
    _ASM("BNE LINE140");
    _ASM("LINE180:");
    _ASM("SUBS R4, R4, #1");
    _ASM("BNE LOOPSTART");
    _ASM("ADD SP, SP, #8");
    _ASM("ADD SP, SP, R3, LSL #2");
    _ASM("LDMIA SP!, {R1-R8}");
    _ASM("MOV PC, LR");
}
```

HP-200LX / 8086 Assembly

```
var
    ary:array[0..8] of byte;
    cnt:word;

begin
    asm
        MOV     WORD PTR CNT,1000
@06:  MOV     DX,8
        LEA     BX,ARY
        MOV     CX,0
        MOV     SI,0
@00:  CMP     DX,SI
        JZ      @05
        INC     SI
        MOV     [BX+SI],DL
@01:  INC     CX
        MOV     DI,SI
@02:  DEC     DI
```

```
JZ @00
MOV AL,[BX+SI]
SUB AL,[BX+DI]
JZ @04
JNC @03
NEG AL
@03: XOR AH,AH
ADD AX,DI
SUB AX,SI
JNZ @02
@04: DEC BYTE PTR [BX+SI]
JNZ @01
DEC SI
JNZ @04
@05: DEC WORD PTR CNT
JNZ @06
MOV CNT,CX
end;
writeln(cnt);
end.
```

=====

COMET Virtual Educational Machine / CASL Assembly

```
BGN START
LEA GR0,8
ST GR0,DIM
LEA GR0,0
LEA GR1,0
L00 CPA GR1,DIM
JZE L05
LEA GR1,1,GR1
LD GR3,DIM
ST GR3,ARY,GR1
L01 ADD GR0,ONE
ST GR1,TMP
LD GR2,TMP
L02 LEA GR2,-1,GR2
JZE L00
LD GR3,ARY,GR1
SUB GR3,ARY,GR2
JZE L04
JPZ L03
EOR GR3,FFH
LEA GR3,1,GR3
L03 ST GR2,TMP
ADD GR3,TMP
ST GR1,TMP
SUB GR3,TMP
JNZ L02
L04 LD GR3,ARY,GR1
LEA GR3,-1,GR3
ST GR3,ARY,GR1
JNZ L01
LEA GR1,-1,GR1
JNZ L04
L05 EXIT
ONE DC 1
FFH DC #FFFF
DIM DS 1
TMP DS 1
ARY DS 9
END
```

(Result in GR0)

=====

COMP-X Virtual Educational Machine / CAP-X Assembly

```
:START:32
BGN:LAI :0:8
:ST :0:DIM
:LAI :0:0
:LAI :1:0
L00:ST :1:TMP
:LD :3:TMP
:SUB :3:DIM
:JNZ :3:L06
:JC :3:L05
L06:LAI :1:1:1
:LD :3:DIM
:ST :3:ARY:1
L01:ADD :0:ONE
:ST :1:TMP
:LD :2:TMP
L02:LAI :2:255:2
:JNZ :2:L07
:JC :3:L00
L07:LD :3:ARY:1
:SUB :3:ARY:2
:JNZ :3:L08
:JC :3:L04
L08:JC :2:L03
:EOR :3:FFH
:LAI :3:1:3
L03:ST :2:TMP
:ADD :3:TMP
:ST :1:TMP
:SUB :3:TMP
:JNZ :3:L02
L04:LD :3:ARY:1
:LAI :3:255:3
:ST :3:ARY:1
:JNZ :3:L01
:LAI :1:255:1
:JNZ :1:L04
L05:WRITE:0:10
:HJ :0:BGN
ONE:CONST:1
FFH:CONST:FFFF
DIM:RESV :1
TMP:RESV :1
ARY:RESV :9
:END :BGN
```

=====

C64 / 6502 Assembly

```
LDA #8
STA DIM
LDX #0
L00: CPX DIM
BEQ L05
INX
LDA DIM
STA ARY,X
L01: TXA
TAY
INC CTL
BNE L02
INC CTH
L02: DEY
BEQ L00
LDA ARY,X
SEC
SBC ARY,Y
BEQ L04
BCS L03
EOR #$FF
ADC #1
L03: STA TMP
TYA
CLC
ADC TMP
```



```

      STA  TMP
      CPX  TMP
      BNE  L02
L04:   DEC  ARY,X
      BNE  L01
      DEX
      BNE  L04
L05:   RTS
CTL:   DC   0
CTH:   DC   0
DIM:   DS   1
TMP:   DS   1
ARY:   DS   9

```

=====

TI-89 / 68000 Assembly

```

      MOVEM D3-D5, -(A7)
      MOVE.W #1000, D5
L06:   MOVEQ #8, D2
      LEA    ARY(PC), A0
      CLR.W  D3
      CLR.L  D0
L00:   CMP.B D0, D2
      BEQ    L05
      ADDQ.B #1, D0
      MOVE.B D2, 0(A0, D0.W)
L01:   ADDQ.W #1, D3
      MOVE.L D0, D1
L02:   SUBQ.B #1, D1
      BEQ    L00
      MOVE.B 0(A0, D0.W), D4
      SUB.B  0(A0, D1.W), D4
      BEQ    L04
      BCC    L03
      NEG.B  D4
L03:   ADD.B D1, D4
      SUB.B  D0, D4
      BNE    L02
L04:   SUBQ.B #1, 0(A0, D0.W)
      BNE    L01
      SUBQ.B #1, D0
      BNE    L04
L05:   SUBQ.W #1, D5
      BNE    L06
      MOVE.W D3, CNT
      MOVEM  (A7)+, D3-D5
      RTS
CNT:   DS.W  1
ARY:   DS.B  9

```

=====

MK-85 / MK-90 / PDP-11 Assembly

```

      .RADIX 16
      MOV    #64, R5
L06:   CLR    R3
      CLR    R0
L00:   CMP    #8, R0
      BEQ    L05
      INC    R0
      MOVB   #8, ARY(R0)
L01:   INC    R3
      MOV    R0, R1
L02:   DEC    R1
      BEQ    L00
      MOVB   ARY(R0), R2
      MOVB   ARY(R1), R4
      SUB    R2, R4
      BEQ    L04

```

```

      BCC      L03
      NEG      R4
L03:  ADD      R1,R4
      SUB      R0,R4
      BNE      L02
L04:  DECB     ARY(R0)
      BNE      L01
      SOB      R0,L04
L05:  SOB      R5,L06
      MOV      R3,CNT
      RTS
CNT:  .BLKW    1
ARY:  .BLKB    9
```

=====

CC-40 / TI-74 / TI-95 / TMS7000 Assembly

```

      CLR      R10
      CLR      R11
      CLR      R12
      CLR      R14
      CLR      R15
L00:  MOV      %8,A
      CMP      R11,A
      JZ       L05
      INC      R11
      STA      *R11
L01:  INC      R15
      ADC      %0,R14
      MOV      R11,R13
L02:  DEC      R13
      JZ       L00
      LDA      *R13
      MOV      A,R9
      LDA      *R11
      SUB      R9,A
      JZ       L04
      JC       L03
      INV      A
      INC      A
L03:  ADD      R13,A
      SUB      R11,A
      JNZ      L02
L04:  LDA      *R11
      DEC      A
      STA      *R11
      JNZ      L01
      DJNZ     R11,L04
L05:  RETS
```

(Array in internal RAM)

=====

PC-1360 / PC-1402 / ESR-H SC61860 Assembly

```

      RA
      LP 8
      EXAM
      RA
      LP 9
      EXAM
      LII 0
L00:  LIA 8
      LP 0
      CPMA
      JRZ L05
      INCI
      EXAB
      LDM
```

```
ADIA 10
STP
EXAB
EXAM
L01: LP 0
LDM
LP 1
EXAM
INCK
JRNZ L02
INCL
L02: DECJ
JRZ L00
LP 0
LDM
ADIA 10
STP
LDM
EXAB
LP 1
LDM
ADIA 10
STP
LDM
LP 3
SBM
JRZ L04
JRNCL L03
ADM
EXAB
SBM
L03: LP 1
LDM
LP 3
ADM
LP 0
LDM
LP 3
SBM
JRNZ L02
L04: LP 0
LDM
ADIA 10
STP
SBIM 1
JRNZ L01
DECI
JRNZ L04
L05: RTN
```

(Array in internal RAM)

=====

PC-E500S / ESR-L SC62015 Assembly

```
MV (00H),8
MV (ECH),0
MV I,0
L00: CMP (00H),(ECH)
JRZ L05
INC (ECH)
MV (BP+00H),(00H)
L01: INC I
MV (EDH),(ECH)
L02: DEC (EDH)
JRZ L00
MV A,(BP+00H)
SUB A,(PX+00H)
JRZ L04
JRNCL L03
XOR A,FFH
INC A
L03: ADD A,(EDH)
SUB A,(ECH)
```

```
JRNZ L02
L04: DEC (BP+00H)
JRNZ L01
DEC (ECH)
JRNZ L04
L05: RTN
```

(Array in internal RAM)

PC-1500A / LH5801 Assembly

```
LDI YH,8
LDI XL,0
L00: LDA YH
CPA XL
BZS L05
INC XL
LDI UH,HI(ARY)
LDI UL,LO(ARY)
LDA XL
ADR U
LDA YH
STA (U)
L01: LDA XL
STA YL
ADI (CNT),1
BZR L02
ADI (CNT+1),1
L02: DEC YL
BZS L00
LDI UH,HI(ARY)
LDI UL,LO(ARY)
LDA XL
ADR U
LDA (U)
STA XH
LDI UH,HI(ARY)
LDI UL,LO(ARY)
LDA YL
ADR U
LDA XH
SEC
SBC (U)
BZS L04
BCS L03
EAI FFH
INC A
L03: REC
ADC YL
CPA XL
BZR L02
L04: LDI UH,HI(ARY)
LDI UL,LO(ARY)
LDA XL
ADR U
ADI (U),-1
BZR L01
DEC XL
BZR L04
L05: RTN
CNT: DW 0
ARY: DS 9
```

PC-E220 / SC7852 Assembly
PC-G850S / PC-G850V / PC-G850VS / TI-83+ SE / Z80 Assembly

ORG 0100H

```
LD HL,0400H
LD (CNT),HL
LD IY,CNT
L06: LD BC,0
LD D,0
LD IX,0
L00: LD A,8
CP C
JR Z,L05
INC C
LD HL,ARY
ADD HL,BC
LD (HL),A
L01: INC IX
LD E,C
L02: DEC E
JR Z,L00
LD HL,ARY
ADD HL,BC
LD A,(HL)
LD HL,ARY
ADD HL,DE
SUB (HL)
JR Z,L04
JR NC,L03
NEG
L03: ADD A,E
SUB C
JR NZ,L02
L04: LD HL,ARY
ADD HL,BC
DEC (HL)
JR NZ,L01
DEC C
JR NZ,L04
L05: DEC (IY)
JR NZ,L06
DEC (IY+1)
JR NZ,L06
LD (CNT),IX
RET
CNT: DW 0
ARY: DS 9
END
```

(SC7852 not T state compatible with Z80: +1 normally / +2 if CB/DD/ED/FD prefixed)

=====

FX-8000G / µPD1007 Assembly

```
LD R0,8*2
LD R1,2
LDM R3..R4,0
LD R73,0
L00: TSB R73,R0
JMP Z,L05
AD R73,2
PST AS,R73
LD I64,R0
L01: ADM R3..R4,1
LD R2,R73
LD R74,R2
L02: SB R74,2
JMP Z,L00
PST AS,R74
LD R2,I64
PST AS,R73
SB R2,I64
JMP Z,L04
JMP NC,L03
CMP R2
L03: AD R2,R74
SB R2,R73
JMP NZ,L02
```

```
L04: PST    AS,R73
      SB     I64,R1
      JMP    NZ,L01
      SB     R73,2
      JMP    NZ,L04
L05: RTN
```

(Array in internal RAM)

=====

FX-7500G / µPD1037 Assembly
SF-9700 / µPD3055 Assembly
SF-9350 / SF-R20 / HD62076 Assembly

```
ARY   ORG    &H7D26
      EQU    &HC000
      LD     R64,8
      LDM    R3..R4,0
      LD     R0,0
L00:   TSB    R0,R64
      SJMP   Z,L05
      AD     R0,1
      LDW    IX,ARY-1
      ADW    IX,R0
      ST     +(IX),R64
L01:   ADM    R3..R4,1
      LD     R65,R0
L02:   SB     R65,1
      SJMP   Z,L00
      LDW    IX,ARY
      ADW    IX,R0
      LD     R5,(IX)+
      LDW    IX,ARY
      ADW    IX,R65
      LD     R66,(IX)+
      SB     R5,R66
      SJMP   Z,L04
      SJMP   NC,L03
      CMP    R5
L03:   AD     R5,R65
      LD     R66,R0
      SB     R5,R66
      SJMP   NZ,L02
L04:   LDW    IX,ARY
      ADW    IX,R0
      SB     (IX),1
      SJMP   NZ,L01
      SB     R0,1
      SJMP   NZ,L04
L05:   RTN
```

=====

FX-890P / Z-1GRA / 8086 Assembly

```
      ORG    2000H
      MOV    WORD PTR CNT,1000
L06:   MOV    DX,8
      LEA    BX,ARY
      MOV    CX,0
      MOV    SI,0
L00:   CMP    DX,SI
      JZ     L05
      INC    SI
      MOV    [BX+SI],DL
L01:   INC    CX
      MOV    DI,SI
L02:   DEC    DI
      JZ     L00
      MOV    AL,[BX+SI]
```

```

SUB AL,[BX+DI]
JZ L04
JNC L03
NEG AL
L03: XOR AH,AH
ADD AX,DI
SUB AX,SI
JNZ L02
L04: DEC BYTE PTR [BX+SI]
JNZ L01
DEC SI
JNZ L04
L05: DEC WORD PTR CNT
JNZ L06
MOV CNT,CX
IRET
CNT DW 0
ARY DB 9 DUP (?)
END
```

```

10 DEFSEG=0
20 CALL &H2000
30 PRINT PEEK(&H2044)+PEEK(&H2045)*256
```

FX-9860G SD / FX-9860G II / SH-3 Assembly

```

.SECTION P,CODE
.EXPORT _QBENCH

_QBENCH:
MOV.L #100000,R7
L06: MOV.B #8,R3
MOV.L ARY,R0
MOV.B #0,R1
MOV.B #0,R4
L00: CMP/EQ R1,R3
BT L05
MOV.B R6,@(R0,R1)
ADD #1,R1
MOV R3,R6
L01: ADD #1,R4
MOV R1,R2
L02: DT R2
BT L00
MOV.B @(R0,R2),R5
CMP/EQ R6,R5
BT L04
SUBC R6,R5
BF L03
NEG R5,R5
L03: ADD R2,R5
CMP/EQ R1,R5
BF L02
L04: DT R6
BF L01
DT R1
MOV.B @(R0,R1),R6
BF L04
L05: DT R7
BF L06
RTS
NOP
ARY: .DATA.L ARX

.SECTION B,DATA
ARX: .RES.B 9
.END
```

PB-1000 / HD61700 Assembly

```
-----
      ORG    &H7000
      START L07
      DS     2
      DS     9
L07:  LDW    $7,1000
L06:  LD      $2,8
      PRE    IZ,&H7002
      LDW    $3,0
      LD      $0,$31
L00:  SBC     $2,$0
      JR      Z,L05
      AD      $0,$30
      ST      $2,(IZ+$0)
L01:  ADW     $3,$30
      LD      $1,$0
L02:  SB      $1,$30
      JR      Z,L00
      LD      $5,(IZ+$0)
      LD      $6,(IZ+$1)
      SB      $5,$6
      JR      Z,L04
      JR      NC,L03
      CMP     $5
L03:  AD      $5,$1
      SB      $5,$0
      JR      NZ,L02
L04:  SB      (IZ+$0),$30
      JR      NZ,L01
      SB      $0,$30
      JR      NZ,L04
L05:  SBW     $7,$30
      JR      NZ,L06
      PRE     IZ,&H7000
      STW     $3,(IZ+$31)
      RTN
```

(Built-in assembler doesn't support all addressing modes)

=====

PB-2000C / HD61700 Assembly

```
{ $c+,d-,f-,i-,r-,s-,t-,v- }
var
  ary:array[0..8] of byte;
  cnt:word;

begin
  inline(
    PST IE,0
    LDW $7,1000
@06:  LD $2,8
      PRE IZ,>ARY
      LDW $3,0
      LD $0,#0
@00:  SBC $2,#2
      JR Z,@05
      AD $0,#1
      ST $2,(IZ+#2)
@01:  ADW $3,#1
      LD $1,#2
@02:  SB $1,#1
      JR Z,@00
      LD $5,(IZ+#2)
      LD $6,(IZ+$1)
      SB $5,$6
      JR Z,@04
      JR NC,@03
      CMP $5
@03:  AD $5,$1
      SB $5,#2
      JR NZ,@02
```



```
@04: SB      (IZ+#2), $30
      JR      NZ, @01
      SB      $0, #1
      JR      NZ, @04
@05: SBW     $7, #1
      JR      NZ, @06
      PRE     IZ, >CNT
      STW     $3, (IZ+#0)
      PST     IE, &H18);
      writeln(cnt);
end.
```

=====

PB-1000 / PB-Forth

```
8 CONSTANT RR
0 VARIABLE SS
0 VARIABLE XX
0 VARIABLE YY
HERE RR 1+ ALLOT CONSTANT AA

: RCLAA @ AA + C@ ;
: STAAA @ AA + C! ;

: NQCORE
0 SS !
0 XX !
BEGIN
1 XX +! RR XX STAAA
BEGIN
1 SS +!
XX @ YY !
BEGIN YY @ 1 > WHILE
-1 YY +!
XX RCLAA YY RCLAA - DUP
0 = SWAP ABS XX @ YY @ - = OR IF
0 YY !
BEGIN XX RCLAA 1 - DUP XX STAAA 0 = WHILE
-1 XX +!
REPEAT
THEN
REPEAT
YY @ 1 = UNTIL
RR XX @ = UNTIL
;

: NQUEENS
( STARTTIMER )
NQCORE
." S=" SS @ .
( DISPLAYTIMER ) CR
;
```

=====

PB-2000C / DL-Pascal ROM Card 1.2

```
{ $c+, d-, f-, i-, r-, s-, t-, v- }
var
  a: array[0..8] of shortint;
  s, n: word;

procedure qbench;
label
  l0, l1, l2, l3;
var
  x, y, r, t: shortint;
begin
  r:=8;
  s:=0;
  x:=0;
```

```

l0: if x=r then exit;
    inc(x);
    a[x]:=r;
l1: inc(s);
    y:=x;
l2: dec(y);
    if y=0 then goto l0;
    t:=a[x]-a[y];
    if t=0 then goto l3;
    if x-y<>abs(t) then goto l2;
l3: dec(a[x]);
    if a[x]<>0 then goto l1;
    dec(x);
    if x<>0 then goto l3;
end;

begin
  inline(PST IE,0);
  for n:=1 to 100 do qbench;
  inline(PST IE,&H18);
  writeln(s);
end.

```

Compiling speed [Turbo x2.2]: 2.4 [1.1] sec

Additional Batch Interpreter Test

```

:0
@a=a+1
@if a>0 then goto 0

```

Execution speed [Turbo x2.2]: 438 [950] loops/min

(Turbo Pascal 5.0 compatible native compiler)

=====

PB-2000C / Prolog ROM Card OM-51P

```

qbench: -
  move([1,2,3,4,5,6,7,8],[],S),
  write(S).
move([],S,S).
move(L,Qs,S):-
  sel(Q,L,R),
  safe(Q,1,Qs),
  move(R,[Q|Qs],S).
sel(X,[X|T],T).
sel(X,[H|T],[H|T1]):-
  sel(X,T,T1).
safe(_,-,[]).
safe(Q,D,[Q1|Qs]):-
  /= (abs(Q-Q1),D),
  D1 is D+1,
  safe(Q,D1,Qs).

```

Execution speed [Turbo x2.2]: 65.4 [30.1] sec

Approximated comparability correction:

```

  65.4 [30.1] sec x 876 Nodes
----- = 170 [78] sec
    336 Nodes

```

Additional LIPS Test

```

reverse([],[]).

```

```
reverse([H|T],L):-
  reverse(T,R),
  append(R,[H],L).
append([],L,L).
append([H|T],L,[H|U]):-
  append(T,L,U).
rev:-
  reverse([1,2,3,4,5,6,7,8,9,0,
           1,2,3,4,5,6,7,8,9,0,
           1,2,3,4,5,6,7,8,9,0],_),!.
r:-
  rev,fail.
r.
lips:-
  r,r,r,r,r,r,r,r,r,r.
```

Execution speed [Turbo x2.2]: 74.4 [34.3] sec

5000 Logical Inferences
----- = 67 [146] LIPS
74.4 [34.3] sec

(DEC-10 Prolog compatible bytecode compiler)

=====

AI-1000 / Lisp

```
(defun qbench ()
  (prog ()
    (setq r 8)
    (setq a (make-array (1+ r)))
    (setq s 0)
    (setq x 0)
  loop0
    (if (= x r) (return s))
    (setf (aref a (setq x (1+ x))) r)
  loop1
    (setq s (1+ s))
    (setq y x)
  loop2
    (if (zerop (setq y (1- y))) (go loop0))
    (if (zerop (setq z (- (aref a x) (aref a y)))) (go loop3))
    (if (/= (- x y) (abs z)) (go loop2))
  loop3
    (unless (zerop (setf (aref a x) (1- (aref a x)))) (go loop1))
    (unless (zerop (setq x (1- x))) (go loop3))))
```

(MacLISP dialect (lexically scoped (tail-call optimized (lisp-1 (bytecode compiler)))))

=====

Credits to all helpers.

pb2000"@"imail.de

Edited: 16 Oct 2014, 12:23 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-9845 used to produce footage for "Wargames" the movie

Posted by [John](#) on 6 Feb 2007, 7:16 a.m.

Hi,

I know the HP-9845 barely passes as an 'HP Calculator', but I just wanted to pass on this snippet of 1983 movie folklore to the HP community.

Browse to <http://www.imsai.net/movies/wargames.htm> and look for the text '9845'.

Quote "Of course, the big technical achievement for the film was the 50,000 or so feet of negative we managed to make into over 100,000 feet of print for the big screens in the front of the set. It was a huge undertaking, especially for the time (we built our own film recorders and file management system to control it all). The crew making those "plates" worked in shifts 24 hours a day for 7 months, if I remember correctly. Oh, and the Hewlett Packard computers we used were H-P 9845c's I believe. They were lent to us by H-P because we were lucky enough to have Colin Cantwell, who had a relationship with H-P and was a consultant to them, as the lead designer of the missile scenarios displayed on those big screens. Colin did an incredible job and it is because of him that I looked so good."

Cool!

Regards,
John

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Fast and Accurate Trigonometric Functions on the HP-17BII

Posted by [Gerson W. Barbosa](#) on 12 Jan 2007, 5:17 p.m.

Each of the following sets of equations below constitutes a complete trigonometric functions implementation for the HP-17BII Financial Calculator. The choice for a particular set will depend on the user's preference. Some sets of equations allow for fast solving time, but with input ranges limited to -90 to 90 degrees for the sine function and 0 to 180 degrees for the cosine function (however all valid input ranges are always allowed for the inverse functions). When the inverse trigonometric functions are missing in the equation set they still may be solved iteratively. The answers are always accurate to at least eleven significant figures all through the specified input ranges.

The table below may be of help when choosing the set of equations:

Equation TAN set Solving time (s)	SIN, TAN ASIN input range Solving (degrees) time (s)	COS input range (degrees)	ASIN, ACOS input range	ATAN input range	Equation Verification time (s)	SIN, COS Solving time (s)
(1) 1.4	-90 to 90 6 (**)	0 to 180			3.2; 6.9	0.8
(2) 2.8	-90 to 90 15 (**)	0 to 180			3.0; 6.1	1.5
(3) 4.7	-999999999999 to 999999999999 18 (**)		-1 to 1 * (all)	-9.999999999999E499 to 9.999999999999E499 * (all)	6.4; 15.2	2.1
(4) 4.6	-999999999999 to 999999999999 3.1				6.7; 15.6	2.1
(5) 2.1	-999999999999 to 999999999999 2.5				6.7; 13.2	1.1

(*) Best performance obtained when providing good initial estimates
(**) Typical (random initial guesses)

A good balance between verification and solving times is obtained with the equation set number 1. Despite the longer solving times, especially for the inverse functions, some users might prefer the equation set number 2 because it is more compact. The equation set number 3 should be an option when full range is required. The benefits of the fast solving time in the equations sets number 4 and 5 are annihilated by their long verification times.

1 - SHORT EQUATIONS (MINIMAX POLYNOMIALS):

SIN=L(Y:X*(5.81776417331E-3+L(X2:SQ(X)))*(-3.2818376137E-8+G(X2)*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20)))))*(3-4*SQ(G(Y)))

COS=L(Y:(90-X)*(5.81776417331E-3+L(X2:SQ(90-X)))*(-3.2818376137E-8+G(X2)*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20)))))*(3-4*SQ(G(Y)))

TAN=(L(Y:X*(5.81776417331E-3+L(X2:SQ(X)))*(-3.2818376137E-8+G(X2)*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20)))))*(3-4*SQ(G(Y)))/L(Y:(90-ABS(X))*(5.81776417331E-3+L(X2:SQ(90-ABS(X)))*(-3.2818376137E-8+G(X2)*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20)))))*(3-4*SQ(G(Y))))

2 - SHORT EQUATIONS (TAYLOR'S SERIES):

SIN=L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))

COS=L(Y:SIGMA(N:0:5:1:(-1)^N*((90-X)*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))

TAN=(L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))/L(Y:SIGMA(N:0:5:1:(-1)^N*((90-ABS(X))*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y))))

3 - IF-LESS FULL-RANGE EQUATIONS (TAYLOR'S SERIES):

SIN=0*L(X:(-1)^(L(Q:IP(L(X:MOD(X:360))/90))+FACT(G(Q)))*((180*(G(Q)-IP(G(Q)/2)))-X))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))

COS=0*L(X:(-1)^(L(Q:IP(L(X:MOD(90-X:360))/90))+FACT(G(Q)))*((180*(G(Q)-IP(G(Q)/2)))-X))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))

TAN=(0*L(X:(-1)^(L(Q:IP(L(X:MOD(L(T:X):360))/90))+FACT(G(Q)))*((180*(G(Q)-IP(G(Q)/2)))-X))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))/(0*L(X:(-1)^(L(Q:IP(L(X:MOD((90-G(T)):360))/90))+FACT(G(Q)))*((180*(G(Q)-IP(G(Q)/2)))-X))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y))))

4 - FULL-RANGE EQUATIONS (TAYLOR'S SERIES):

SIN=0*L(X:IF(L(X:MOD(X:360))<90:X:IF(X<270:180-X:X-360)))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))

COS=0*L(X:IF(L(X:MOD(90-X:360))<90:X:IF(X<270:180-X:X-360)))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))

TAN=(0*L(X:IF(L(X:MOD(X:360))<90:X*L(T:1):IF(X<270:(180-X)*SQ(L(T:-1)):X-360)*L(T:1)))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y)))/(0*L(X:IF(ABS(L(X:(90-X)*G(T)))<90:X:(180-ABS(X))*SGN(X)))+L(Y:SIGMA(N:0:5:1:(-1)^N*(X*PI/540)^(2*N+1)/FACT(2*N+1)))*(3-4*SQ(G(Y))))

$$1)^{N*(X*\pi/540)^{(2*N+1)/FACT(2*N+1))}*(3-4*SQ(G(Y))))$$

$$ASIN=0*(L(S1:SGN(X))+L(X:ABS(X))+L(X:IF(X<>1:L(F:1)*X/SQRT(1-SQ(X)):0*(L(F:2))+X))+L(X:IF(X<1:L(S2:1)*X+L(K1:0):0*(L(K1:\pi/2)+L(S2:-1))+1/X))+L(X:IF(X>2-SQRT(3):0*L(K2:\pi/6)+(X*SQRT(3)-1)/(X+SQRT(3)):X+L(K2:0))))+(G(K1)+G(S2)*(SIGMA(N:0:11:1:(-1)^N*X^{(2*N+1)/(2*N+1))+G(K2)))*G(F)*G(S1)*180/\pi$$

$$ACOS=0*(L(X:IF(X<>-1:L(F:1)*SQRT((1-X)/(1+X)):0*(L(F:-2))+X))+L(S1:SGN(X))+L(X:ABS(X))+L(X:IF(X<1:L(S2:1)*X+L(K1:0):0*(L(K1:\pi/2)+L(S2:-1))+1/X))+L(X:IF(X>2-SQRT(3):0*L(K2:\pi/6)+(X*SQRT(3)-1)/(X+SQRT(3)):X+L(K2:0))))+(G(K1)+G(S2)*(SIGMA(N:0:11:1:(-1)^N*X^{(2*N+1)/(2*N+1))+G(K2)))*G(F)*G(S1)*360/\pi$$

$$ATAN=0*(L(S1:SGN(X))+L(X:ABS(X))+L(X:IF(X<1:L(S2:1)*X+L(K1:0):0*(L(K1:\pi/2)+L(S2:-1))+1/X))+L(X:IF(X>2-SQRT(3):0*L(K2:\pi/6)+(X*SQRT(3)-1)/(X+SQRT(3)):X+L(K2:0))))+(G(K1)+G(S2)*(SIGMA(N:0:11:1:(-1)^N*X^{(2*N+1)/(2*N+1))+G(K2)))*G(S1)*180/\pi$$

5 - FULL-RANGE EQUATIONS (MINIMAX POLYNOMIALS):

$$SIN=L(Y:L(X:IF(L(X:MOD(X:360))<90:X:IF(X<270:180-X:X-360)))*(5.81776417331E-3+L(X2:SQ(X))*(-3.2818376137E-8+G(X2))*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20))))*(3-4*SQ(G(Y))))$$

$$COS=L(Y:L(X:IF(L(X:MOD(90-X:360))<90:X:IF(X<270:180-X:X-360)))*(5.81776417331E-3+L(X2:SQ(X))*(-3.2818376137E-8+G(X2))*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20))))*(3-4*SQ(G(Y))))$$

$$TAN=(L(Y:L(X:IF(L(X:MOD(X:360))<90:X*L(T:1):IF(X<270:(180-X)*SQ(L(T:-1):(X-360)*L(T:1))))*(5.81776417331E-3+L(X2:SQ(X))*(-3.2818376137E-8+G(X2)*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20))))*(3-4*SQ(G(Y))))/(L(Y:L(X:IF(ABS(L(X:(90-X)*G(T)))<90:X:(180-ABS(X))*SGN(X)))*(5.81776417331E-3+L(X2:SQ(X))*(-3.2818376137E-8+G(X2)*(5.55391606E-14+G(X2)*(2.0935E-26*G(X2)-4.47566E-20))))*(3-4*SQ(G(Y))))$$

$$ASIN=0*(L(S1:SGN(X))+L(X:ABS(X))+L(X:IF(X<>1:L(F:1)*X/SQRT(1-SQ(X)):0*(L(F:2))+X))+L(X:IF(X<1:L(S2:1)*X+L(K1:0):0*(L(K1:\pi/2)+L(S2:-1))+1/X))+L(X:IF(X>2-SQRT(3):0*L(K2:\pi/6)+(X*SQRT(3)-1)/(X+SQRT(3)):X+L(K2:0))))+(G(K1)+G(S2)*(X*(1+L(X2:SQ(X))*(-0.3333333333+G(X2)*(0.19999999631+G(X2)*(-0.1428565387+G(X2)*(0.1110748114+G(X2)*(0.0641264*G(X2)-0.08991517)))))))+G(K2))*G(F)*G(S1)*180/\pi$$

$$ACOS=0*(L(X:IF(X<>-1:L(F:1)*SQRT((1-X)/(1+X)):0*(L(F:-2))+X))+L(S1:SGN(X))+L(X:ABS(X))+L(X:IF(X<1:L(S2:1)*X+L(K1:0):0*(L(K1:\pi/2)+L(S2:-1))+1/X))+L(X:IF(X>2-SQRT(3):0*L(K2:\pi/6)+(X*SQRT(3)-1)/(X+SQRT(3)):X+L(K2:0))))+(G(K1)+G(S2)*(X*(1+L(X2:SQ(X))*(-0.3333333333+G(X2)*(0.19999999631+G(X2)*(-0.1428565387+G(X2)*(0.1110748114+G(X2)*(0.0641264*G(X2)-0.08991517)))))))+G(K2))*G(F)*G(S1)*360/\pi$$

$$ATAN=0*(L(S1:SGN(X))+L(X:ABS(X))+L(X:IF(X<1:L(S2:1)*X+L(K1:0):0*(L(K1:\pi/2)+L(S2:-1))+1/X))+L(X:IF(X>2-SQRT(3):0*L(K2:\pi/6)+(X*SQRT(3)-1)/(X+SQRT(3)):X+L(K2:0))))+(G(K1)+G(S2)*(X*(1+L(X2:SQ(X))*(-0.3333333333+G(X2)*(0.19999999631+G(X2)*(-0.1428565387+G(X2)*(0.1110748114+G(X2)*(0.0641264*G(X2)-0.08991517)))))))+G(K2))*G(S1)*180/\pi$$

Note: SIGMA in the equations above is the summation character ([ALPHA] [WXYZ] [OTHER] [MORE])

The table below is a demonstration of both the accuracy and ease of use of these equations in the HP-17BII Solver. These results were obtained with the equation set number 1 (assuming the equations are ordered in the list as SIN, COS, TAN).

Keystrokes	Display	Calculation
9 [X] [SIN] [EXIT] [v] [CALC] [STO] [X] [COS] [EXIT] [v] [CALC] [STO] [X] [TAN] [STO] [TAN] [X] [EXIT] [v] [CALC] [STO] [COS] [X] [EXIT] [v] [CALC] [STO] [SIN] [X]	SIN=0.15643446504 COS=9.99996272745E-1 TAN=1.74549998555E-2 X=9.99996272745E-1 X=1.56434385713E-1 X=8.99999539825	sin(9) cos(sin(9)) tan(cos(sin(9))) atan(tan(cos(sin(9)))) acos(atan(tan(cos(sin(9))))) asin(acos(atan(tan(cos(sin(9)))))

This is Mike Sebastian's [Forensic Result](#) for that set of equations. Sets 2, 3 and 5 return 9.00000194973, 9.00005188184 and 9.00000229472, respectively.

The fastest results were obtained with the use Minimax Polynomials, as expected. Range reduction has also been important in achieving such accurate results in relatively short time. When Taylor's Series are used solving times increase because fifty per cent more terms are required in the polynomials and the coefficients have to be calculated on the fly. However this allows for more compact equations. For more details refer to [Fast and Accurate Trigonometric Functions on the HP-12C Platinum](#)

Note:

In the minimax polynomial equations above, the basic sine equation could be replaced with this one:

$$\text{SIN} = L(Y : X * (\text{PI} / 900 + L(X2 : \text{SQ}(X)) * (-7.088769242\text{E} - 9 + G(X2) * (4.3187175\text{E} - 15 - 1.2504\text{E} - 21 * G(X2)))) * (\text{SQ}(G(Y)) * (16 * \text{SQ}(G(Y)) - 20) + 5))$$

It's slightly shorter and almost as fast (0.9 seconds) and gives answers accurate to eleven digits on the HP-17BII (maximum absolute error: 1.3E-12). The constant 5.81776417331E-3 in the original equations can be replaced with PI/540 at the cost of a few tens of milliseconds in solving time.

Edited: 14 Aug 2007, 6:35 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous \]](#) [\[Next \]](#)

Eighth Order Linear Equation Solver for the hp 33s

Posted by [Palmer O. Hanson, Jr.](#) on 6 Oct 2006, 12:24 a.m.

I started with a TI-59 linear equation solution program by Henrik Ohlsson which appeared in the 81-2 issue of the Swedish newsletter Programbiten. Henrik's program expanded the capability of the TI-59 from order 8 to order 16.

I translated Henrik's program for the Durabrand 828 as a means of demonstrating the capability of that little low cost machine. I was able to fit a sixth order linear equation solution into the Durabrand. The translation was essentially a straightforward, brute force modification from AOS to EOS.

I translated the Durabrand program into a sixth order linear equation solution for the hp 33s. The resulting hp 33s program was essentially a brute force translation of Henrik's program from the TI-59 for the Durabrand and from the Durabrand for the hp 33s. The resulting sixth order linear equation solution was documented as Article 676. The hp 33s program wasn't pretty but it was functional. Only the 26 data registers were used. Register arithmetic was used throughout (no use of the stack in RPN and no use of parentheses or equal signs in ALG) so the the program as written would run in either RPN or ALG mode. That wasn't necessary but it was something that I had wanted to demonstrate.

Next, I found a way to use the statistics registers to expand the capability of the hp 33s program to solve a seventh order set of linear equations. Only limited changes were required to make the expansion. The changes were posted in a September 17 message "Expansion to seventh order" in the Forum.

Finally, I expanded the solution to solve an eighth order set of linear equations. To do that Henrik's method required 33 data registers. The combination of the data registers and the statistics registers of the hp 33s yields only 32 registers. I found a modification to Henrik's method which eliminated the need for one register. The sixth and seventh order solutions had used data registers Y and Z as scratch pads in order to implement the register arithmetic type solution. I had to free up those registers to yield the 32 registers needed for my modification to Henrik's method. To do that I changed substantial parts of the program to use the RPN stack. The resulting program follows.

I won't be doing an expansion to ninth order. Henrik's method would require five more data registers to do that.

Program Listing:

```

N0001    LBL N
N0002    CLVARS
N0003    CLSum
N0004    INPUT N
N0005    STO A
N0006    0
N0007    STO N
N0008    8
N0009    STO C
00001    LBL 0
00002    0
00003    STO G
00004    RCL A
00005    STO B
00006    1

```

```
00007 ST0+ B
00008 8
00009 ST0 F
00010 RCL D
00011 IP
00012 ST0 E
00013 ST0 D
00014 1
00015 ST0+ D
00016 RCL E
00017 x=0?
00018 GT0 C
A0001 LBL A
A0002 XEQ S
A0003 +/-
B0001 LBL B
B0002 ENTER
B0003 ENTER
B0004 RCL F
B0005 XEQ T
B0006 R Down
B0007 RCL (i)
B0008 x
B0009 RCL C
B0010 XEQ T
B0011 R Down
B0012 ST0+ (i)
B0013 R Down
B0014 R Down
B0015 1
B0016 ST0+ C
B0017 ST0+ F
B0018 R Down
B0019 DSE B
B0020 GT0 B
B0021 RCL A
B0022 ST0 B
B0023 ST0- C
B0024 1
B0025 ST0+ B
B0026 ST0- C
B0027 DSE E
B0028 GT0 A
C0001 LBL C
C0002 XEQ S
C0003 ST0 E
C0004 RCL C
C0005 XEQ T
C0006 RCL (i)
C0007 ST0+ E
C0008 1
C0009 ST0- B
D0001 LBL D
D0002 XEQ S
D0003 ENTER
D0004 1
D0005 ST0+ C
D0006 R Down
D0007 RCL C
D0008 XEQ T
D0009 R Down
D0010 RCL (i)
D0011 +
D0012 0
D0013 ST0 (i)
D0014 1
D0015 ST0- C
D0016 R Down
D0017 R Down
D0018 RCL C
D0019 XEQ T
D0020 R Down
D0021 ST0 (i)
D0022 RCL E
D0023 ST0/ (i)
D0024 1
D0025 ST0+ C
D0026 DSE B
D0027 GT0 D
D0028 RCL D
D0029 IP
```

```
D0030  ST0 E
D0031  DSE E
D0032  GT0 I
D0033  GT0 H
I0001  LBL I
I0002  7
I0003  ST0 C
I0004  RCL A
I0005  ST0 B
E0001  LBL E
E0002  1
E0003  ST0+ C
E0004  ST0+ G
E0005  RCL C
E0006  XEQ T
E0007  0
E0008  x<> (i)
F0001  LBL F
F0002  ENTER
F0003  ENTER
F0004  1
F0005  ST0+ C
F0006  R Down
F0007  RCL F
F0008  XEQ T
F0009  R Down
F0010  RCL (i)
F0011  +/-
F0012  x
F0013  RCL C
F0014  XEQ T
F0015  R Down
F0016  RCL (i)
F0017  +
F0018  0
F0019  ST0 (i)
F0020  R Down
F0021  RCL G
F0022  ST0- C
F0023  R Down
F0024  RCL C
F0025  XEQ T
F0026  R Down
F0027  ST0 (i)
F0028  RCL G
F0029  ST0+ C
F0030  R Down
F0031  1
F0032  ST0+ F
F0033  R Down
F0034  R Down
F0035  DSE B
F0036  GT0 F
F0037  RCL A
F0038  ST0- F
F0039  ST0 B
F0040  DSE E
F0041  GT0 E
F0042  RCL G
F0043  ST0- C
F0044  1
F0045  ST0+ C
G0001  LBL G
G0002  RCL F
G0003  XEQ T
G0004  RCL (i)
G0005  RCL C
G0006  XEQ T
G0007  R Down
G0008  ST0 (i)
G0009  RCL F
G0010  XEQ T
G0011  0
G0012  ST0 (i)
G0013  1
G0014  ST0+ C
G0015  ST0+ F
G0016  DSE B
G0017  GT0 G
H0001  LBL H
H0002  DSE A
```

```
H0003 GT0 0
H0004 7
H0005 ST0 i
H0006 RCL D
H0007 IP
H0008 ST0+ i
H0009 1000
H0010 ST0/ i
H0011 8
H0012 ST0+ i
J0001 LBL J
J0002 RCL (i)
J0003 STOP
J0004 ISG i
J0005 GT0 J
J0006 0
J0007 1/x
J0008 STOP
S0001 LBL S
S0002 0.01
S0003 ST0+ D
S0004 RCL D
S0005 FIX 2
S0006 STOP
S0007 FIX 9
S0008 RTN
T0001 LBL T
T0002 ST0 i
T0003 27
T0004 -
T0005 x<0?
T0006 RTN
T0007 R Down
T0008 1
T0009 ST0+ i
T0019 RTN
```

To use the program press XEQ N and see the prompt N? in the display. Enter the order of the solution desired (n = 2 to 6) and press R/S. 1.01 will appear in the display indicating that the machine is ready to accept the A(1,1) element of the matrix Enter the A(1,1) element and press R/S. 1.02 will appear in the display indicating that the machine is ready to accept the A(1,2) element of the matrix. Proceed as before to enter the A(1,2) element and see the prompt for the A(1,3) element, and so on. When the column element of the prompt is n+1 enter the corresponding element of the vector. When the last element of the vector has been entered the machine will stop with 8.000000000 in the Y register display and the first element of the solution, X(1), in the X register display. Press R/S to see X(2), and so on. When all of elements of the solution vector have been displayed the message DIVIDE BY 0 will be displayed. The elements of the solution can be recalled starting with register H followed by subsequent registers as needed by the order of the system.. .

If you run the problem with the matrix as a 8x8 sub-Hilbert and the vector all ones you will get the following results:

Exact	hp 33s	Relative Error
-72	-79.85564194	0.10674
2520	2752.76793665	0.09237
-27720	-29976.5053369	0.08140
138600	148685.022829	0.07276
-360360	-384064.400797	0.06578
504504	534783.269832	0.06002
-360360	-380245.814081	0.05518
102960	108218.029350	0.05107

for a mean relative error of 7.316E-02. Hendrik's program running on the TI-59 yields a mean relative error of 3.331E-02. The ML-02 program running on the TI-59 yields a mean relative error of 2.212E-03. The matrix solution in the HP-28 yield a mean relative error of 1.801E-03. .

Some notes on the program:

1. With Henrik's program on the TI-59 the calculations after the entry of an element required a significant amount of time so that the user was always waiting on the machine. For example, in a fifth order solution when the vector

element is entered in response to the prompt 3.06 the TI-59 will run for nine seconds in fast mode or fifteen seconds in normal mode before the next prompt (4.01) appears. The much higher calculating speed of the hp 33s yields much shorter wait times.

- 2. Henrik used an INV Dsz command in his TI-59 program. I had to synthesize the equivalent in the hp 33s which yielded the curious code at steps D0021 through D0033..
- 3. The T routine provides the methodology for easily adding the statistics registers immediately after the data registers, essentially skipping the i register.
- 4. The STO/ (i) command at step D0023 will cause a divide by zero error for some systems. Consider the following problem:

2	-1	-3	1	1
-2	1	1	3	1
2	-1	0	-1	1
-3	2	-1	0	1

where the solution 5.0, 8.25, 0.5, 0.75 can be readily obtained with the TI-59 ML-02 program, with the H-41 Math Pac and with others. If you attempt to solve the problem with this hp 33s program you will get a "Divide by 0" error after the A(2,3) element is entered. If you start the program over again and enter the first row and then the third row you will again get the "Divide by 0" error after the A(2,3) element is entered. However, if you start the program one more time and enter the first row, followed by the fourth row, the second row and the third row in that order you will see the correct solution. The point of all this is that if you get the "Divide by 0" error after the entry of an element you must try another arrangement of the rows.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

Sixth Order Linear Equation Solver for the hp 33s

Posted by [Palmer O. Hanson, Jr.](#) on 16 Sept 2006, 10:37 p.m.

This program is a translation for the hp 33s of a translation for the Durabrand 828 of a TI-59 program by Henrik Ohlsson which appeared in the 81-2 issue of the Swedish newsletter Programbiten. Henrik's program expanded the linear equation solution capability of the TI-59 from order 8 to order 16. The translations were essentially brute force command by command from AOS to EOS to hp 33s language. The resulting hp 33s program isn't pretty but it is functional. Henrik's program was written for fast mode on the TI-59 which meant that subroutines were not allowed. I modified Henrik's program to run in normal mode and perform the data entry routine as a subroutine in order to save enough space to fit the program in the 828. Because only register arithmetic was used in the hp 33s version (no use of the stack in RPN and no use of parentheses or equal signs in ALG) the hp 33s program will run as written in either RPN or ALG mode.

Program Listing:

```
N0001 LBL N
N0002 CLVARS
N0003 INPUT N
N0004 STO A
N0005 STO G
N0006 1
N0007 STO+ G
N0008 0
N0009 STO N
N0010 9
N0011 STO C
O0001 LBL O
O0002 0
O0003 STO H
O0004 RCL G
O0005 STO B
O0006 9
O0007 STO F
O0008 RCL D
O0009 IP
O0010 STO E
O0011 STO D
O0012 1
O0013 STO+ D
O0014 RCL E
O0015 x=0?
O0016 GTO C
```

A0001 LBL A
A0002 XEQ S
A0003 RCL Z
A0004 +/-
A0005 STO Z
B0001 LBL B
B0002 RCL F
B0003 STO i
B0004 RCL (i)
B0005 STO Y
B0006 RCL Z
B0007 STOx Y
B0008 RCL C
B0009 STO i
B0010 RCL Y
B0011 STO+ (i)
B0012 1
B0013 STO+ C
B0014 STO+ F
B0015 DSE B
B0016 GTO B
B0017 RCL G
B0018 STO B
B0019 STO- C
B0020 DSE E
B0021 GTO A
C0001 LBL C
C0002 XEQ S
C0003 RCL Z
C0004 STO E
C0005 RCL C
C0006 STO i
C0007 RCL (i)
C0008 STO+ E
C0009 1
C0010 STO- B
D0001 LBL D
D0002 XEQ S
D0003 1
D0004 STO+ C
D0005 RCL C
D0006 STO i
D0007 RCL (i)
D0008 STO+ Z
D0009 0
D0010 STO (i)
D0011 1
D0012 STO- C
D0013 RCL C
D0014 STO i
D0015 RCL Z
D0016 STO (i)
D0017 RCL E
D0018 STO/ (i)

D0019 1
D0020 STO+ C
D0021 DSE B
D0022 GTO D
D0023 RCL D
D0024 IP
D0025 STO E
D0026 DSE E
D0027 GTO I
D0028 GTO H
I0001 LBL I
I0002 8
I0003 STO C
I0004 RCL A
I0005 STO B
E0001 LBL E
E0002 1
E0003 STO+ C
E0004 STO+ H
E0005 RCL C
E0006 STO i
E0007 RCL (i)
E0008 STO Z
E0009 0
E0010 STO (i)
F0001 LBL F
F0002 1
F0003 STO+ C
F0004 RCL F
F0005 STO i
F0006 RCL (i)
F0007 STO Y
F0008 RCL Z
F0009 +/-
F0010 STOx Y
F0011 RCL C
F0012 STO i
F0013 RCL (i)
F0014 STO+ Y
F0015 0
F0016 STO (i)
F0017 RCL H
F0018 STO- C
F0019 RCL C
F0020 STO i
F0021 RCL Y
F0022 STO (i)
F0023 RCL H
F0024 STO+ C
F0025 1
F0026 STO+ F
F0027 DSE B
F0028 GTO F
F0029 RCL A

F0030 STO- F
F0031 STO B
F0032 DSE E
F0033 GTO E
F0034 RCL H
F0035 STO- C
F0036 1
F0037 STO+ C
G0001 LBL G
G0002 RCL F
G0003 STO i
G0004 RCL (i)
G0005 STO Y
G0006 RCL C
G0007 STO i
G0008 RCL Y
G0009 STO (i)
G0010 RCL F
G0011 STO i
G0012 0
G0013 STO (i)
G0014 1
G0015 STO+ C
G0016 STO+ F
G0017 DSE B
G0018 GTO G
H0001 LBL H
H0002 1
H0003 STO- G
H0004 DSE A
H0005 GTO O
H0006 9
H0007 STO i
H0008 RCL H
H0009 STO+ i
H0010 1000
H0011 STO/ i
H0012 9
H0013 STO+ i
J0001 LBL J
J0002 RCL (i)
J0003 STOP
J0004 ISG i
J0005 GTO J
J0006 GTO K
J0007 STOP
S0001 LBL S
S0002 0.01
S0003 STO+ D
S0004 RCL D
S0005 FIX 2
S0006 STOP
S0007 FIX 9
S0008 STO Z

S0009 RTN

To use the program press XEQ N and see the prompt N? in the display. Enter the order of the solution desired (n = 2 to 6) and press R/S. 1.01 will appear in the display indicating that the machine is ready to accept the A(1,1) element of the matrix. Enter the A(1,1) element and press R/S. 1.02 will appear in the display indicating that the machine is ready to accept the A(1,2) element of the matrix. Proceed as before to enter the A(1,2) element and see the prompt for the A(1,3) element, and so on. When the column element of the prompt is n+1 enter the corresponding element of the vector B. When the last element of the vector has been entered the machine will stop with the first element of the solution, X(1), in the display. Press R/S to see X(2), and so on. When all of elements of the solution vector have been displayed the message NONEXISTENT will be displayed.. .

If you run the problem with the matrix as a 6x6 sub-Hilbert and the vector all ones you will get the following answer:

Exact	hp 33s	Relative Error
-42	-41.999994975	1.196E-07
840	839.999760819	2.847E-07
-5040	-5039.99824184	3.488E-07
12600	12599.9953562	3.686E-07
-13860	-13859.9948996	3.680E-07
5544	5543.99801446	3.581E-07

for a mean relative error of 3.080E-07. Hendrik's program running on the TI-59 yields a mean relative error of 8.79E-06. The Durabrand translation yielded a mean relative error of 7.03E-06. The ML-02 program running on the TI-59 yields a mean relative error of 2.90E-06. The matrix routine in the Math Pak of the HP-41 yields a mean relative error of 3.8E-04. The matrix solution in the HP-28 yield a mean relative error of 9.99 E-06. The simultaneous equation solver on the TI-85 yields a mean relative error of 3.012E-07..

Historical notes: With Hendrik's program on the TI-59 the calculations after the entry of an element required a significant amount of time so that the user was always waiting on the machine. For example, in a fifth order solution when the third vector element is entered in response to the prompt 3.06 the TI-59 would run for nine seconds in fast mode or fifteen seconds in normal mode before the next prompt (4.01) appears. The much higher calculating speed of the hp 33s yields much shorter wait times.

You can see Henrik's program by going to Viktor Toth's site (www.rskey.org/), selecting the Texas Instruments part of the library, and going to page 16 of the V8N6 issue of TI PPC Notes. I did the translation for the Durabrand 828 when Richard Nelson and I were assessing the capabilities of that machine. The listing of the translation for the Durabrand 828 follows: Program 0 is the main program. Program 1 is the data entry subroutine. Since I don't know how to print the data entry triangle I have used the {} symbol in place of the data entry triangle in the listing.

Program 0 (388 steps)

```

Mcl : "N=" {} ? -> B : B + 1 -> H : 9 -> D :
Lbl 0 : 0 -> I : H -> C : 9 -> G : Int E -> F : F + 1 -> E : F = 0 => Goto 3 :
Lbl 1 : Prog 1 : - A -> A :
Lbl 2 : A x A[G] + A[D] -> A[D] : D + 1 -> D : G + 1 -> G : Dsz C : Goto 2 :
H -> C : D - H -> D : Dsz F : Goto 1 :
Lbl 3 : Prog 1 : A + A[D] -> F : C - 1 -> C :
Lbl 4 : Prog 1 : D + 1 -> D : A + A[D] -> A : 0 -> A[D] : D - 1 -> D : A / F -> A[D] : D + 1 -> D : Dsz C : Goto 4 :
Int E -> F : Dsz F : Go to 9 :
Goto 8 :
Lbl 9 : 8 -> D : B -> C :
Lbl 5 : D + 1 -> D : I + 1 -> I : A[D] -> A : 0 -> A[D] :
Lbl 6 : D + 1 -> D : A[D] - A x A[G] -> Z : 0 -> A[D] : D - I -> D : Z -> A[D] : D + I -> D : G + 1 -> G : Dsz C :
Goto 6 :
G - B -> G : B -> C : Dsz F : Goto 5 :
D - I + 1 -> D :

```

```
Lbl 7 : A[G] -> A[D] : 0 -> A[G] : D + 1 -> D : G + 1 -> G : Dsz C : Goto 7 :  
Lbl 8 : H - 1 -> H : Dsz B : Goto 0 :  
J{}
```

Program 1 (10 step subroutine)

```
E + .01 -> E {} ? -> A
```

The Durabrand 828 does NOT provide alphanumeric prompting as described in paragraph 4.9 (page 47) of the manual. For example the manual indicates that if the user enters the command "N="?"->A then the machine will display N=? and wait for an input. The fx-7000G does that. The Durabrand does not. To see the prompt with the Durabrand one has to insert the "Display Results" triangle between the = and the ? . The calculator will then stop with X= in the display. The user will have to press EXE to see the question mark and press EXE again after the input value has been entered.

To use the program select Program 0, press EXE and see the prompt N= in the display. Press EXE a second time and see a question mark in the display. Enter the order of the solution desired (n = 2 to 6) and press EXE. 1.01 will appear in the display indicating that the machine is ready to accept the A(1,1) element of the matrix Press EXE and see a question mark in the display. Enter the A(1,1) element and press EXE. 1.02 will appear in the display indicating that the machine is ready to accept the A(1,2) element of the matrix. Proceed as before to enter the A(1,2) element and see the prompt for the A(1,3) element, and so on. When the column element of the prompt is n+1 enter the corresponding element of the vector. When the last element of the vector has been entered the machine will stop with the first element of the solution, X(1), in the display. Press AC, press Alpha J and press EXE to see the same value for X(1). Then press Alpha K followed by EXE to see X(2), Alpha L followed by EXE to see X(3) and so on as needed up to Alpha O to see the rest of the solution. .

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Fast and Accurate Trigonometric Functions on the HP-12C Platinum

Posted by [Gerson W. Barbosa](#) on 6 Aug 2006, 3:10 p.m.

The program

001 - 2	023 - CHS	045 - g x<=y	067 - /
089 - g GT0 266	024 - g x<=y	046 - g GT0 017	068 - +
002 - STO 6	025 - g GT0 031	047 - Rv	069 - RCL 8
090 - 2	026 - 2	048 - x<>y	070 - *
003 - Rv	027 - *	049 - STO 8	071 - ENTER
091 - STO 6	028 - x<>y	050 - g x^2	072 - g x^2
004 - RCL .0	029 - -	051 - ENTER	073 - 4
092 - Rv	030 - ENTER	052 - ENTER	074 - *
005 - /	031 - Rv	053 - ENTER	075 - CHS
093 - RCL .0	032 - ENTER	054 - RCL .2	076 - 3
006 - x<>y	033 - g x^2	055 - *	077 - +
094 - /	034 - g SQRT	056 - RCL .3	078 - *
007 - STO 4	035 - ENTER	057 - -	079 - RCL 6
095 - CHS	036 - ENTER	058 - *	080 - 2
008 - x<>y	037 - 9	059 - RCL .4	081 - g x<=y
096 - 9	038 - 0	060 - +	082 - g GT0 087
009 - ENTER	039 - -	061 - *	083 - Rv
097 - 0	040 - g x=0	062 - RCL .5	084 - g x=0
010 - g x^2	041 - g GT0 047	063 - -	085 - g GT0 114
098 - +	042 - Rv	064 - *	086 - g GT0 109
011 - g SQRT	043 - 9	065 - RCL .1	087 - Rv
099 - g GT0 006	044 - 0	066 - 3	088 - x<>y
012 - 3			
100 - RCL .0			
013 - 6			
101 - /			
014 - 0			
102 - STO 4			
015 - g x<=y			
103 - x<>y			
016 - g GT0 120			
104 - STO 7			
017 - Rv			
105 - 1			
018 - Rv			
106 - STO 6			
019 - 9			
107 - RCL 4			
020 - 0			
108 - g GT0 009			
021 - g x<=y			
109 - CLX			
022 - g GT0 026			
110 - STO 6			
111 - x<>y	152 - 0	193 - g x<=y	234 - RCL .7
275 - Rv	153 - *	194 - g GT0 197	235 - +
112 - RCL 4	154 - RCL .0	195 - 1/x	236 - *
276 - g GT0 000	155 - *	196 - 9	237 - RCL .8
113 - g GT0 095	156 - g GT0 266	197 - 0	238 - -
114 - x<>y			
115 - RCL 4			

116 -	x<>y	157 -	x<>y	198 -	RCL .1	239 -	*
117 -	/	158 -	STO 4	199 -	*	240 -	RCL .9
277 -	1.745329252	159 -	x<>y	200 -	STO 6	241 -	+
118 -	RCL 7	160 -	2	201 -	x<>y	242 -	*
288 -	EEX	161 -	STO 7	202 -	2	243 -	3
119 -	g GT0 267	162 -	Rv	203 -	ENTER	244 -	1/x
289 -	2	163 -	1	204 -	3	245 -	-
120 -	x<>y	164 -	x<>y	205 -	g SQRT	246 -	*
290 -	CHS	165 -	+	206 -	STO 9	247 -	1
121 -	EEX	166 -	g x=0	207 -	-	248 -	+
291 -	STO .1	167 -	g GT0 269	208 -	x<>y	249 -	RCL 9
122 -	1	168 -	g LSTx	209 -	g x<=y	250 -	*
292 -	2.1	169 -	1	210 -	g GT0 222	251 -	RCL 8
123 -	2	170 -	x<>y	211 -	ENTER	252 -	+
295 -	EEX	171 -	-	212 -	ENTER	253 -	RCL 6
124 -	g x<=y	172 -	g LSTx	213 -	RCL 9	254 -	g x=0
296 -	26	173 -	1	214 -	*	255 -	x<>y
125 -	g GT0 146	174 -	+	215 -	1	256 -	x<>y
298 -	CHS	175 -	/	216 -	-	257 -	-
126 -	Rv	176 -	g SQRT	217 -	x<>y	258 -	RCL 5
299 -	STO .2	177 -	g GT0 184	218 -	RCL 9	259 -	*
127 -	x<>y	178 -	x<>y	219 -	+	260 -	RCL 7
300 -	4.47569	179 -	STO 4	220 -	/	261 -	*
128 -	/	180 -	x<>y	221 -	3	262 -	RCL .1
307 -	EEX	181 -	1	222 -	0	263 -	/
129 -	g LSTx	182 -	STO 7	223 -	RCL .1	264 -	RCL .0
308 -	20	183 -	x<>y	224 -	*	265 -	*
130 -	x<>y	184 -	ENTER	225 -	STO 8	266 -	RCL 4
310 -	CHS	185 -	g x^2	226 -	x<>y	267 -	x<>y
131 -	g INTG	186 -	g SQRT	227 -	STO 9	268 -	g GT0 000
311 -	STO .3	187 -	g x=0	228 -	g x^2	269 -	RCL 7
132 -	*	188 -	g GT0 266	229 -	ENTER	270 -	g GT0 151
312 -	5.55391606	189 -	/	230 -	ENTER	271 -	1
133 -	g x<=y	190 -	STO 5	231 -	ENTER	272 -	g GT0 274
322 -	EEX	191 -	1	232 -	RCL .6	273 -	RCL .1
134 -	CHS	192 -	g LSTx	233 -	*	274 -	STO .0
323 -	14						
135 -	+						
325 -	CHS						
136 -	g GT0 019						
326 -	STO .4						
137 -	x<>y						
327 -	3.281837614						
138 -	STO 4						
338 -	EEX						
139 -	x<>y						
339 -	8						
140 -	ENTER						
340 -	CHS						
141 -	g x^2						
341 -	STO .5						
142 -	1						
342 -	.0784						
143 -	-						
347 -	CHS						
144 -	g x=0						
348 -	STO .6						
145 -	g GT0 150						
349 -	.110351						
146 -	CHS						
356 -	STO .7						
147 -	g SQRT						
357 -	.142838						
148 -	/						
364 -	STO .8						
149 -	g GT0 181						
365 -	.199999839						
150 -	x<>y						
375 -	STO .9						
151 -	9						
376 -	g GT0 271						

Lines 277 through 376 are optional. This is an initialization routine that loads the constants used by the program and sets the angular to degrees. This has to be done before running the program for the first time or whenever the registers in the range R.1 through R.9 are accidentally cleared or changed (unlikely possibility since at least three key presses would be required: STO . 0..9; f CLEAR REG). In case

these lines
are not keyed in the constants should be entered manually:

```

R.1:  1.745329252E-02    (pi/180)

R.2:          2.1E-26
R.3:          4.47569E-20
R.4:          5.55391606E-14
R.5:          3.281837614E-08    (sine polynomial coefficients)

R.6:          -0.0784
R.7:          0.110351
R.8:          0.142838
R.9:          0.199999839    (arctangent polynomial coefficients)

```

The next arctangent polynomial coefficient, 1/3, is built into the program (lines 243 and 244). The registers 4 through .0 are used by the program but they can be overwritten anytime. The registers 0 through 3 and the financial registers are always available to the user.

Usage

```

SIN:  R/S (*)
COS:  g GT0 090 R/S
TAN:  g GT0 100 R/S
ASIN: g GT0 137 R/S
ACOS: g GT0 157 R/S
ATAN: g GT0 178 R/S

```

```

DEG:  g GT0 271 R/S
RAD:  g GT0 273 R/S

```

g GT0 277 R/S: If available, this loads all constants needed by the program.

(*) When just coming from an error condition due to an invalid argument, use **g GT0 001 R/S** instead, as the calculator will not return to program line 000.

Another way to set the angular mode to DEGREES is just storing 1 in register R.0: **1 STO .0**. Likewise, **.9 1/x STO .0** sets the angular mode to GRADS. As a convenience, register .1 (pi/180) may be used to convert to and from RADIANS.

Input ranges

SIN, COS, TAN: $|x| < 1E12$ (degrees)

ASIN, ACOS: $|x| \leq 1$

ATAN: $|x| \leq 9.999999999E49$

Notes:

1) Although in DEGREES mode the answers of SIN, COS and ATAN functions are accurate at least to 11 significant figures all through the valid ranges, in RADIANS mode a loss of accuracy occurs for larger absolute values of x.

2) The program will return 9.999999E99 for input arguments with absolute values equal or greater than 1E50.

Arguments out of the allowed ranges will cause the program to return **Error 0**. Arguments for which the tangent function are not defined, such as 90 degrees, will also cause an **Error 0** message to appear.

3) Pay attention to what appears to be a bug in the HP-12C Platinum: negating a number just entered by pressing the CHS (change sign) key does not work the way it used to on the classic HP-12C. For instance:

Keystrokes	Display	
45 ENTER	45.00000000	
CHS	-45.00000000	The number displayed by the calculator is
negative,		
5	5.	
+	50.00000000	yet it has been interpreted as a positive number.
15 ENTER	15.00000000	
CHS	-15.00000000	we've just entered 15,
15) and press CHS		but then we remember we want to compute sin(-
R/S	0.258819045	then R/S. Wrong result, the calculator shows
sin(15) instead!		

So, either we enter the correct number the first time or we remember to press ENTER after pressing CHS:

15 CHS R/S	-0.258819045	sine(-15)
15 ENTER	15.00000000	
CHS ENTER	-15.00000000	
R/S	-0.258819045	sine(-15), as expected.

Running Times

SIN, COS: 2.5 s $\left(\begin{array}{l} |x| \leq 180 \\ |x| \leq 180 \end{array} \right)$
TAN: 4.9 s
ASIN, ACOS, ATAN: 2.6 s

Sample Calculations

First set the display to FIX 9 and DEG modes if not already set:

f 9; g GT0 271 R/S

1) asin(acos(atan(tan(cos(sin(9)))))):

Keystrokes	Display
9	9.
R/S	0.156434465
g GT0 090 R/S	0.999996273
g GT0 100 R/S	0.017455000
g GT0 178 R/S	0.999996273
g GT0 157 R/S	0.156434568
g GT0 137 R/S	9.000005947

2) $\sin^2(30) + \cos^2(30)$:

Notice that the stack register X is always preserved, so there's no need to store intermediate results in simple chain calculations (one-level only), as in this example.

Keystrokes	Display
30 R/S	0.500000000
g x ²	0.250000000
30 g GT0 090 R/S	0.866025404
g x ²	0.750000000
+	1.000000000

3) sin(-1234567.8) + acos(-0.77)

Keystrokes	Display
1234567.8 CHS	-1,234,567.8
R/S	-0.790155012
.77 CHS g GT0 157 R/S	140.3538889
+	139.5637338

4) tan(-6.2 rad)/asin(0.15 rad)

Set the angular mode to RADIANS:

g GT0 273 R/S

Keystrokes	Display
6.2 CHS g GT0 100 R/S	0.083377715
0.15 g GT0 137 R/S	0.150568273
/	0.553753545
g GT0 271 R/S	0.553753545 ; back to DEG mode

Accuracy comparison with other HP calculators

Sin(x):

x (deg)	HP-12CP	HP-50G (2006)	HP-15C (1982)	HP-35 (1972)
0.000000	0.000000000000E+00	0.000000000000E+00	0.0000000000E+00	0.0000000000E+00
0.000010	1.74532925200E-07	1.74532925199E-07	1.745329252E-07	1.745000000E-07
0.000110	1.91986217720E-06	1.91986217719E-06	1.919862177E-06	1.919800000E-06
0.022000	3.83972426005E-04	3.83972426004E-04	3.839724260E-04	3.839723910E-04
3.330000	5.80867495981E-02	5.80867495977E-02	5.808674960E-02	5.808674960E-02
14.44000	2.49366025116E-01	2.49366025115E-01	2.493660251E-01	2.493660250E-01
25.55000	4.31298587031E-01	4.31298587031E-01	4.312985870E-01	4.312985871E-01
36.66000	5.97065256389E-01	5.97065256389E-01	5.970652564E-01	5.970652561E-01
47.77000	7.40452782677E-01	7.40452782677E-01	7.404527827E-01	7.404527828E-01
58.88000	8.56086728293E-01	8.56086728292E-01	8.560867283E-01	8.560867285E-01
69.99000	9.39632912698E-01	9.39632912698E-01	9.396329127E-01	9.396329127E-01
81.11000	9.87986852782E-01	9.87986852778E-01	9.879868528E-01	9.879868527E-01
88.88000	9.99808950037E-01	9.99808950038E-01	9.998089500E-01	9.998089499E-01
89.99900	9.99999999848E-01	9.99999999848E-01	9.999999998E-01	9.999999998E-01
89.99990	9.99999999998E-01	9.99999999998E-01	1.0000000000E+00	1.0000000000E+00
90.00000	1.000000000000E+00	1.000000000000E+00	1.0000000000E+00	1.0000000000E+00

Tan(x):

x (deg)	HP-12CP	HP-50G (2006)	HP-15C (1982)	HP-35 (1972)
0.000000	0.000000000000E+00	0.000000000000E+00	0.0000000000E+00	0.0000000000E+00
0.000010	1.74532925200E-07	1.74532925199E-07	1.745329252E-07	1.745000000E-07
0.000110	1.91986217721E-06	1.91986217720E-06	1.919862177E-06	1.919800000E-06
0.022000	3.83972454310E-04	3.83972454309E-04	3.839724543E-04	3.839724542E-04

3.330000	5.81849926709E-02	5.81849926706E-02	5.818499267E-02	5.818499260E-02
14.44000	2.57500649118E-01	2.57500649118E-01	2.575006491E-01	2.575006490E-01
25.55000	4.78047179798E-01	4.78047179799E-01	4.780471798E-01	4.780471798E-01
36.66000	7.44291588299E-01	7.44291588300E-01	7.442915883E-01	7.442915880E-01
47.77000	1.10168657755E+00	1.10168657755E+00	1.101686578E+00	1.101686578E+00
58.88000	1.65641139080E+00	1.65641139081E+00	1.656411391E+00	1.656411391E+00
69.99000	2.74598611677E+00	2.74598611678E+00	2.745986117E+00	2.745986119E+00
81.11000	6.39316645081E+00	6.39316645080E+00	6.393166451E+00	6.393166426E+00
88.88000	5.11504299317E+01	5.11504299320E+01	5.115042993E+01	5.115042860E+01
89.99900	5.72957795071E+04	5.72957795073E+04	5.729577951E+05	5.729655162E+04
89.99990	5.72957795128E+05	5.72957795130E+05	5.729577951E+05	5.730193057E+05
90.00000	Error 0	TAN Error	9.999999999E+99	9.999999999E+99

ArcTan(x) :

x	HP-12CP	HP-50G (2006)	HP-15C (1982)	HP-35 (1972)
0.00000	0.00000000000E+00	0.00000000000E+00	0.000000000E+00	0.000000000E+00
0.00011	6.30253572098E-03	6.30253572102E-03	6.302535721E-03	6.302535688E-03
0.15500	8.81073298598E+00	8.81073298598E+00	8.810732986E+00	8.810732984E+00
0.26795	1.50000431708E+01	1.50000431708E+01	1.500004317E+01	1.500004317E+01
0.41421	2.24998257819E+01	2.24998257819E+01	2.249982578E+01	2.249982579E+01
0.57735	2.99999884329E+01	2.99999884324E+01	2.999998843E+01	2.999998843E+01
0.77700	3.78472067673E+01	3.78472067673E+01	3.784720677E+01	3.784720676E+01
0.88800	4.16050764577E+01	4.16050764579E+01	4.160507646E+01	4.160507646E+01
1.00000	4.50000000000E+01	4.50000000000E+01	4.500000000E+01	4.500000000E+01
1.22200	5.07054870168E+01	5.07054870169E+01	5.070548702E+01	5.070548702E+01
1.48880	5.61114572284E+01	5.61114572285E+01	5.611145723E+01	5.611145722E+01
2.11100	6.46526573517E+01	6.46526573518E+01	6.465265735E+01	6.465265735E+01
4.88800	7.84378235856E+01	7.84378235855E+01	7.843782359E+01	7.843782360E+01
7.55500	8.24600068273E+01	8.24600068272E+01	8.246000683E+01	8.246000679E+01
99.9990	8.94270555733E+01	8.94270555733E+01	8.942705557E+01	8.942705555E+01
3333333	8.99999828113E+01	8.99999828113E+01	8.999998281E+01	8.999998281E+01
1.1E+12	8.99999999999E+01	8.99999999999E+01	9.000000000E+01	9.000000000E+01

Of course, the HP-12C Platinum cannot display all those twelve significant digits simultaneously. They were obtained in most cases by multiplying the number by a power of 10 and taking the fractional part of the result. As we can notice in these examples, all answers are accurate to at least 11 digits, so the results displayed by the HP-12C will most of the time match those displayed by all scientific HP 10-series calculators (HP-10C, HP-11C and HP-15C). On some occasions there might be a difference of one unit in the last significant displayed digit. This happens because these calculators round the answers to 10 digits and on rare occasions there may be a difference on one unit in the last digit of the mantissa, as we can see in the following examples (DEG mode assumed):

1)	HP-12CP	HP-15C	
Keystrokes	Display	Keystrokes	Display
32.888	32.888	32.888	32.888
R/S	0.542998588	SIN	0.542998589
f CLEAR PREFIX	5429985885	f CLEAR PREFIX	5429985885
sin(32.888) to 16 digits: 0.5429985884655923			
rounding to 12 digits: 0.542998588466			
HP-12CP answer: 0.542998588466			
'4'	displayed 9-digit answer: 0.542998588	; correct on the HP-12CP, because of the leading	
'6'	10-digit mantissa: 5429985885	; correct on the HP-12CP, because of the leading	
rounding to 10 digits: 0.5429985885			
'5'	displayed 9-digit answer: 0.542998589	; correct on the HP-15C, because of the leading	

In this case, the answer of the HP-15C is right, according to its rounding method.

2) HP-12CP		HP-15C	
Keystrokes	Display	Keystrokes	Display
34.444	34.444	34.444	34.444
R/S	0.565600478	SIN	0.565600479
f CLEAR PREFIX	5656004784	f CLEAR PREFIX	5656004785
sin(34.444) to 16 digits: 0.5656004784499306			
rounding to 12 digits: 0.565600478450			
HP-12CP answer: 0.565600478449			
'4'	displayed 9-digit answer: 0.565600478	; correct on the HP-12CP, because of the leading	
'4'	10-digit mantissa: 5656004784	; correct on the HP-12CP, because of the leading	
rounding to 10 digits: 0.5656004784			
leading '4'	displayed 9-digit answer: 0.565600479	; incorrect on the HP-15C, because of the	
leading '4'	10-digit mantissa: 5656004785	; incorrect on the HP-15C, because of the	

In this case, according to its rounding method, the HP-15C should return 0.5656004784. Unless, of course, a slight error in the thirteenth significant digit has turned those '499' into '500' or greater.

Since the HP-12CP does not round the answers to 10 digits it doesn't make much sense to implement this feature in the program. Anyway, this is possible by means of a slight modification:

1) Change the line 268 to

```
268-      g GT0 377
```

2) Key in these lines:

```
377-      ENTER
378-      g x^2
379-      g SQRT
380-      g x=0
381-      g GT0 000
382-      g LN
383-      1
384-      0
385-      g LN
386-      /
387-      g INTG
388-      1
389-      -
390-      1
391-      0
392-      x<>y
393-      y^x
394-      /
395-      g LSTx
396-      x<>y
397-      f RND
398-      x<>y
399-      *
```

This will round the answers to the number of the decimal places of the display plus one. However results with absolute value less than 1E-50 will be rounded to zero.

After this change, let's do the first example again, actually a standard test created by Mike Sebastian to check trigonometric algorithms in calculators, called [Calculator Forensics](#). The angular mode should be set to DEG and the display to FIX 9 mode.

1) $\text{asin}(\text{acos}(\text{atan}(\text{tan}(\text{cos}(\text{sin}(9))))))$:

Keystrokes	Display
9	9.
R/S	0.156434465
g GT0 090 R/S	0.999996273
g GT0 100 R/S	0.017455000
g GT0 178 R/S	0.999996272
g GT0 157 R/S	0.156441660
g GT0 137 R/S	9.000417403

The result exactly matches that of the HP-15C and other HP 10-digit calculators. Because of the difference in the last digit that occasionally may happen, as we have seen, this rounding won't assure the answers will always be the same on both calculators.

Don't forget to revert to the original program by editing the line 268:

```
g GT0 267
f P/R
g GT0 000
f P/R
```

The method

In order to achieve results that accurate under relatively short times, we used minimax polynomial approximations and range reduction. This technique yields maximum accuracy throughout a desired range by approximating a function by means of a polynomial of as low degree as possible. In this case, the sine function is approximated by a 9th-degree polynomial in the range $[0..\pi/6]$ and the arctangent function is approximated by an 11th-degree polynomial in the range $[0..2-\sqrt{3}]$. These are not the full polynomials with all coefficients of all available powers we would obtain by default when calling minimax in most numerical math packages. Instead, by means of an initial work, we have obtained polynomials with odd powers only and with lowest-order coefficients equal to one, more akin to the classical infinite series for these functions.

Programs that approximate trigonometric functions using the plain Taylor series are generally more compact. Nevertheless, to obtain the same accuracy throughout these ranges using Taylor series we would need to evaluate up to 20 percent more terms in the sine polynomial and up to 50 percent more terms in the arctangent polynomial, not to mention that at the end of each iteration a test should be made to verify the accuracy. Furthermore, computing the coefficients on the fly would make the program even slower. Of course, in the older version of the HP-12C calculator, with only 99 programming steps and very few general use storage registers, there were few options left, if any. However, the HP-12C Platinum, with four times as much programming memory and 20 general use registers always available, is quite suitable for this solution. Since there's plenty of memory, the program is not so optimized as it could be, so there's still room for improvement. Anyway, the results obtained so far are better than what we might have expected. In fact, the program turns the HP-12C Platinum into the most accurate HP 10C-series calculator, regarding trigonometric functions.

The following expressions have been used:

$$(i) \quad \sin(3x) = \sin(x) * (3 - 4 \sin^2(x))$$

$$(ii) \quad \sin(x) = x(a_1 + x^2(a_2 + x^2(a_3 + x^2(a_4 + a_5 x^2))))), \quad x \text{ in radians};$$

These are the original coefficients, optimized for the range $[0..pi/6]$:

```
a1: 1.000000000000E+00
a2: -1.666666666666E-01
a3: 8.33333320429E-03
a4: -1.98410347969E-04
a5: 2.74201854577E-06
```

They are multiplied by successive odd powers of $pi/540$ for radians to degrees conversion and to avoid an initial division by three:

```
a1: 5.81776417333E-03
a2: -3.28183761371E-08
a3: 5.55391605878E-14
a4: -4.47566022234E-20
a5: 2.09351175453E-26
```

This yields a maximum absolute error of $8E-14$. As we don't need this level of accuracy, the coefficients used in the program have been slightly modified to use as few digits as possible while keeping the maximum absolute error under $1.2E-12$.

$$(iii) \quad \text{atan}(x) = (pi/6 + \text{atan}((|x| * \text{sqrt}(3) - 1) / (|x| + \text{sqrt}(3)))) * x/|x|, \quad (2 - \text{sqrt}(3)) < |x| \leq 1;$$

$$(iv) \quad \text{arctan}(x) = (pi/2 - \text{arctan}(1/|x|)) * x/|x|, \quad |x| > 1;$$

$$(v) \quad \text{atan}(x) = x(a_1 + x^2(a_2 + x^2(a_3 + x^2(a_4 + x^2(a_5 + a_6 x^2))))), \quad x \text{ in radians};$$

These are the original coefficients, optimized for the range $[0..2-\text{sqrt}(3)]$:

```
a1: 1.000000000000E+00
a2: -3.33333333089E-01
a3: 1.99999829275E-01
a4: -1.42837953536E-01
a5: 1.10350743154E-01
a6: -7.83982197548E-02
```

The maximum absolute error with the original coefficients is less than $5E-12$. Again, in order to minimize the number of digits, the following set of coefficients has been chosen. Although this would in theory give us a maximum absolute error of $7.8E-12$, the maximum observed error is $5E-12$. It is possible the internal floating point errors have compensated the maximum error for better. Also, a small part of the error is due to the use of $1.745329252E-02$ as the angle conversion factor instead of $1.74532925199E-02$. Entering the exact value to 12 digits has not produced any

noticeable
improvement in the overall accuracy though.

a_1 : 1.00000000000E+00
 a_2 : -3.33333333333E-01
 a_3 : 1.9999839000E-01
 a_4 : -1.4283800000E-01
 a_5 : 1.1035100000E-01
 a_6 : -7.8400000000E-02

$$(vi) \quad \cos(x) = \sin(\pi/2 - x)$$

$$(vii) \quad \tan(x) = \sin(x)/\cos(x)$$

$$(viii) \quad \arcsin(x) = \arctan(x/(\sqrt{1 - x^2})) \quad |x| \leq 1;$$

$$(ix) \quad \arccos(x) = 2 * \arctan(\sqrt{(1 - x)/(1 + x)}) \quad 0 < x \leq 1;$$

References

1) For some methods to approximate trigonometric functions, refer to the following paper, by Robin Green of Sony Computer Entertainment America:

Faster Math Functions

http://www.research.scea.com/research/pdfs/RGREENfastermath_GDC02.pdf

2) *HP-71B Minimax Polynomial Fit*, by Valentin Albillo:

<http://membres.lycos.fr/albillo/calc/pdf/DatafileVA016.pdf>

Edited: 30 Dec 2006, 9:15 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

EMU41 Update 2006

Posted by *Christoph Klug* on 15 June 2006, 4:44 p.m.

EMU41 Update 2006

Again Jean-Francois Garnier helpfully adds new features to his nice EMU41 emulator software : Up to now, for advanced applications like handling rom-images, creating own user code modules or machine code development, EMU41 emulated a 8KByte MLDL RAM page and the famous W&W RAM-Box. This virtual RAM-Box consist of two software selected 32 KByte banks = 64KByte RAM.

ERAMCO 128KByte RSU

Now Jean-Francois also emulates the Eramco RSU RAM-Storage-Unit which supports four software selected 32 KByte banks = 128KByte RAM. The operating system for the W&W RAM-Box is stored in a 4KByte block. Similarly the Eramco operation system is stored in a 8KByte block and includes additionally commands for data storing applications and its own text editor. In my experience the 128KByte RSU from Eramco is the greatest "classic" RAM-Box solution designed for the HP41 system. I have also heard rumours about an extended 256 KByte RSU solution (8 software selected banks) but have never seen this. To activate the Eramco 128Kbyte RSU to EMU41, edit the EMU41.INI file and add the following extensions :

8 96 es_rsu.dat ; the 96k RAM = 4 banks of 24k in pages 8 to 13 14 8 rsu_os.rom ; the RSU OS (supporting the multiple banks)

In this case the 8 KByte RSU operating system is addressed to port 14 and port 15. If so, as Jean-Francois explained, it would be possible to override RSU pages by plug in modules. In the following example we place the CCD-Module to port 8 and port 9 :

8 96 es_rsu.dat 14 8 rsu_os.rom 8 8 ccd1b.bin ; override pages 8 & 9

When operating the RSU, remember that Eramco numbers the plug in port pages in an own way : W&W counts from page 8 up to port 15, identical to the the Zenrom page numbering. In contrast to this, Eramco starts with +1 for port 8, -1 for port 9 and ends with +4 for port 14 and -4 for port 15.

RS232 support

Besides this great implementation of the Eramco 128KByte RSU (which works with its own bank switching mechanism) Jean-Francois also adds a virtual HP-IL/RS232 interface device to EMU41, supporting the serial COM port of the PC. Now EMU41 includes the following list of virtual HP-IL devices (taken from the EMU41.INI file) :

[DEVICES] ; DISPLAY LPRTER1 HDRIVE1 ;HDRIVE2 SERIAL1 1,E3 ;SERIAL2 2,E3 ;FDRIVE1 ; insert a LIF DD floppy for correct function. ; Not available for modern PCs. DOSLINK XIL 300 ; HP-IL/PC Interface Card - 700 default ; address / 300 address for WIN2000/NT/XP

The new SERIAL1 device acts like a virtual HP82164A HP-IL/RS232 interface for EMU41 and works as a gateway to your real HP41 handheld. Use this to transfer the X-Register contents, programs, data register blocks, data-files and matrix-files (CCD-Module) between EMU41 and HP41 handheld computer. For your real HP41 you need the IL-Module, EXT-I/O and CCD-Module plus the HP-IL/RS232 interface HP82164A or CMT-RAM-DISC, wired to the serial COM port of the PC.

This new RS232 gateway solution between HP41 and EMU41 is handy for PC configurations that can not use the HP-IL/PC Interface Card : For example notebook- and laptop-PC's and PC's with motherboards without an ISA-slot. As Jean-Francois explained, it would be also possible to add a RS232-to-USB bus interface : With this expanded configuration you get a gateway solution for PC hardware that only supports the USB-Bus.

In contrast to the "classic" HP-IL/PC Interface Card gateway solution, the new RS232 application is limited to bidirectional program- and data transfer. Using the new RS232 gateway you are not able to control real HP-IL devices by EMU41. Controlling real HP-IL devices like the IL-Digitalvoltmeter or the IL-Plotter by EMU41 is only possible by using the HP-IL/PC Interface Card.

HP41 software

In order to use the new RS232 gateway between EMU41 and your real HP41 I create a software pack plus associated user documentation. You find this on the HPCC website. Because HP41 and EMU41 are running with different speeds (depending on the PC), and no COM data buffer exist on PC side for holding the transferred data for EMU41, we have a time critical application. Therefore the correct system configuration and time alignment is needed on PC side.

For data transfer we use the EXT-I/O commands OUTAN and INAN because only from this results a correct and save data transfer for this RS232 application. This two commands transfer a dummy-byte D followed by 7 data-bytes from the Alpha-Register of the sending device to the Alpha-Register of the receiving device. To exchange the 7 data-bytes between Alpha-Register and X-Register we use the (synthetic) commands STO M and RCL M.

For program download from EMU41 to HP41 we use the standard commands OUTP and INP. Because of the critical timing needed for program upload from HP41 to EMU41 we can not use the commands OUTP and INP. Instead we use the advanced commands PEEKB and POKEB from CCD-Module for the transfer of program bytes to the X-Register. Next STO M and RCL M transfer the program bytes to the Alpha-Register, and the commands OUTAN and INAN execute the RS232 transfer. The EMU41 software routine RECVF for receiving program code acts as a self programming program. After the program code transfer is complete, RECVF writes the new .END. position to status register c.

New EMU41 applications

Great thanks to Jean-Francois for expanding EMU41 with advanced RSU storage and RS232 gateway features. Jean-Francois has also added this new RS232 extension to EMU71. This is an encouraging result from a discussion during the last Allschwil meeting (HP-Collection) in Switzerland. In my own opinion EMU41 is the world's leading HP41 emulator software because :

- EMU41 runs on different HP handheld computer- and PC platforms - EMU41 allows the rom-image formats .bin and .rom - EMU41 includes plug in modules and bank switched modules - EMU41 includes HEPAX, MBK-Profiset, RAM-BOX and RSU support - EMU41 includes a gateway for data exchange to other PC applications - EMU41 includes different gateway solutions for interfacing your real HP41 - EMU41 includes some virtual IL devices like printer and mass storage - EMU41 includes virtual tapes for email exchange of data, programs, rom-images - EMU41 is able to control real HP-IL devices

May be this development will result in some more HP41 applications : For example an umbilical cord from EMU41, using RS232 + USB, to the MLDL2000 unit from Meindert Kuipers ?

Christoph Klug Körnerstrasse 47B 31141 Hildesheim Germany

Spring 2006

Edit Message

Delete Message

 Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous \]](#) [\[Next \]](#)

A Label Saving Simultaneous Equation Solution for the 33s

Posted by [Palmer O. Hanson, Jr.](#) on 29 May 2006, 10:19 a.m.

Introduction

The "Solution of Simultaneous Equations" program on pages 15-12 through 15-20 of the *hp 33s user's guide* solves using matrix inversion and matrix multiplication. It includes 136 program lines and eight labels. The program length is not a problem. The use of eight of only 26 available labels is.

A program in the museum catalog "Cramer's Rule for the HP-32S2" by Adam Szporer avoids the lengthy process of matrix inversion. It can be easily moved to the hp 33s. The Szporer program uses 96 program lines and four labels; however, that program does not provide a data entry routine. Rather, the user must enter the elements with a series of STO commands from the keyboard.

The program below also solves using Cramer's rule. It uses the data entry and determinant routines from the *hp 33s user's guide*. The program uses 82 program lines but only four labels.

Program Description

Data entry uses the T routine and the first four lines of of U routine. Those lines are the same as the data entry routine on page 15-13 of the *hp 33s user's guide*. The elements of the matrix is stored in variables A through I. The elements of the vector is stored in variables J through L. The remainder of the U routine calculates the determinant of the matrix and stores the value in variable W. The V routine solves using Cramer's rule. All determinant calculations are performed using the D routine routine in the *hp 33s user's guide*. The elements of the solution vector are stored in variables X, Y and Z. The solution vector is displayed using VIEW commands. An additional R/S after the Z value has been displayed returns the user to the top of the program with the original matrix elements and vector elements in place.

User Instructions

1. Press XEQ T to start the program.
2. Enter the matrix elements in order by columns, followed by the vector elements.
3. When the third vector element has been entered the program will proceed to solve. If the determinant of the input matrix is zero program execution will stop with the message "DIVIDE BY ZERO" in the display.
4. If the determinant of the input matrix is not zero program execution will stop with the X element of the solution vector in the display. Press R/S to see the Y element of the solution vector. Press R/S again to see the Z element of the solution vector.
5. Press R/S again to return to the input routine with the original input in place.
6. To solve a second order system follow the instructions on page 15-18 of the *hp 33s user's guide*.

Program Listing

In addition to the listing below the user must enter the D routine from pages 15-16 and 15-17 of the *hp 33s user's guide*.

```

T0001 LBL T
T0002 1.012
T0003 STO i
U0001 LBL U
U0002 INPUT (i)
U0003 ISG i
U0004 GTO U
U0005 1.00903
U0006 STO i
U0007 21
U0008 STO V
U0009 XEQ D
U0010 STO W
U0011 1/x
V0001 LBL V
V0002 RCL (i)
V0003 STO M
V0004 RCL J
V0005 STO(i)
V0006 1
V0007 STO+ i
V0008 RCL(i)
V0009 STO N
V0010 RCL K
V0011 STO (i)
V0012 1
V0013 STO+ i
V0014 RCL (i)
V0015 STO O
V0016 RCL L
V0017 STO (i)
V0018 XEQ D
V0019 RCL/ W
V0020 RCL V
V0021 STO+ i
V0022 x<>y
V0023 STO (i)
V0024 x<>y
V0025 STO- i
V0026 RCL O
V0027 STO (i)
V0028 1
V0029 STO- i
V0030 RCL N
V0031 STO (i)
V0032 1
V0033 STO- i
V0034 RCL M

```

V0035 STO (i)
V0036 2
V0037 STO- V
V0038 ISG i
V0039 GTO V
V0040 VIEW X
V0041 VIEW Y
V0042 VIEW Z
V0043 GTO T

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP41CY - Replacing the internal +3v lithium cell

Posted by [Christoph Klug](#) on 27 Feb 2006, 4:09 a.m.

HP41CY - Replacing the internal +3V Lithium Cell

The HP41CY, an upgrade of the HP41CX halfnut model, created by W&W Software Products (Germany) includes a turbo speed mode plus an internally 64KByte RAM-Box. Like the W&W RAM-Box which is placed in the card reader shell, the internally RAM chips are buffered by a small +3V CR3025 lithium cell. This additionally +3V supply voltage is only needed, when the four batteries of the HP41CY are removed or when dropped down to low voltage levels.

The RAM consist of several 4KByte pages, needed for the storage of individual user applications like mcode programs, user code programs, data, or images of existing ROM solutions. Furthermore the belonging operating system of the RAM-Box is stored in the RAM itself. Unfortunately from this hardware architecture results a long time problem : When the +6V battery voltage of HP41CY falls down for long time, or when the HP41CY is stored without batteries, the internal +3V lithium cell buffers the RAM contents. When the +3V lithium cell itself drops down, the complete RAM contents would be cleared, than also the operating system is lost. The CR3025 lithium cell would have a typically lifetime duration of 10 years, may be up to 20 years, depending on the main battery state of the HP41CY. Therefore for today HP41CY systems exist a high risk to loose the internal operating system needed for the internal RAM-Box.

Reloading the operating system seems to be a really complex procedure, and special knowledge and plug in tools are needed. Therefore do not wait until your RAM-Box operating system is lost and your HP41CY is a worthless machine. Alternatively it is much easier to replace the +3V lithium cell by a new battery for saving the valuable operating system for much more years :

First switch OFF the handheld than remove the battery block. Now open the HP41CY case by removing the four glued rubber feet and the four screws. Put the face of the forty one down to the workbench and lift up the lower case from the frame. Attention : Do not ignore the thin wires form beeper and turbo switch, may be you have to rotate the lower case. Now you see the internals of the HP41CY, the small bridge board = Clock and X-Memory, below this the additionally RAM-Box board with +3V lithium cell and below this the halfnut main board.

Have a look to the upper region of the halfnut main board (below the display driver), here you find the golden contact area which is needed for interfacing the battery & port assembly. Contact 3 from the left side is ground potential.. Contact 11 from the left side is the +6V potential from the battery. For your orientation - on the left edge of the main board is also placed the big electrolyth condenser. Follow the pcb leads from the two golden contacts 3 and 11 to the next via. Here you are able to solder two wires, for powering the HP41CY by an external +6V DC power supply unit. Check the correct polarity of the external +6V DC power supply unit, than switch ON the HP41CY. Now the handheld works without the lower case and without the belonging battery block Switch OFF the handheld.

Now you are able to remove the old +3V lithium cell, because now the RAM is buffered by the external +6V DC power supply unit. During re-soldering the old cell avoid any shortcut between the +/- contacts and to the RAM-Box hardware. Solder short wires to the connectors of a new CR3025 lithium cell and insulate the metal housing with

plastic tape. Solder the new cell to the RAM-Box board with correct polarity and with the correct mechanical position. Again avoid any shortcuts. Execute a short function test by switching ON the handheld and running CAT 2 : If your work is correct CAT 2 lists –RAMBOX 64a or -RAMBOX 64b, the operating system was not lost during the lithium cell replacement process. Switch OFF the handheld, than remove the wires to the external +6V DC power supply.

Place the lower case shell to the correct mechanical position on the frame above the upper housing – this would be only possible when the lithium cell is in a correct position. Insert the four screws and have lot attention during fixing them to the old screw flight : Do not crack the sensitive screw posts of the upper housing shell. Last insert the battery block and switch ON the HP41CY for an final function test : By executing PG<> you toggle form one 32KByte page to the other 32KByte page of the RAM-Box. Generally operate and store your HP41CY only with a set of new batteries. This will help to extend the lifetime of the replaced +3V lithium cell...

Long live your HP41CY – Christoph Klug

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Some thoughts about possibilities uploading High-End Pioneer and Clamshell ROM images

Posted by [Christoph Giesselink](#) on 20 Feb 2006, 3:27 p.m.

ROMUpload of High-End Pioneers and Clamshell calculators, the serial approach

One of the biggest problems of using calculator emulators is the need of having a ROM image. Because of copyright reasons it's not possible to publish them as file. The good luck is that the calculators have an infrared transmitter LED and most of them are able to execute self written binary code.

1. Transfer methods

1.1 The first approach getting a HP42S ROM image was made by Jean-Francois Garnier using the internal memory scanner by manual transmitting the ROM image, eight bytes each, over the transmitter LED in the redeye format used by the HP82240A/B printer. The bad point is, you need a special receiver to capture the data and be able to transfer the data to a PC. The best common receiver is a HP48 calculator so far. The most worst thing is, that the process is very very slow, we are talking about ~4 hours! More about this in JFG's article [HP-42S: New Facts](#).

1.2 The 2nd approach done by me, based on Jean-Francois one. The only difference is, that a used a machine program transmitting the data in binary and as a whole with no user interaction after starting. The transfer time was reduced to 16 min. plus the time to enter the machine program (10 - 15 min.). But it's still necessary to have the special receiver for the redeye format and in difference to the first approach with enough RAM to hold the complete ROM image. More about this at <http://privat.swol.de/ChristophGiesselink/Emu28/CPROMUPL.ZIP>.

1.3 My 3rd approach is now changing the transfer protocol. Instead of using the redeye format I tried to implement the RS232C protocol with 2400 baud 8,N,1. First without success. The problem in general is that you have to do all timing by software. First the effective CPU speed is not constant, it depend on various factors. Therefore the redeye implementation in the calculators itself are measuring the CPU speed. To keep my implementation simple I don't care about this at the moment. The bigger problem brought me some sleepless nights is, that code running in RAM differs very much from the one running in ROM from timing aspects. Furthermore, all commands executed in RAM have a different timing when they are executed on an even or odd address. My solution for this problem is executing the timing critical parts in the display RAM. Display RAM is directly connected to the Saturn bus like the ROM and has exactly the same timing. This implementation is now independent if the code begins on an even or odd address.

But has the receiver problem being solved? Sorry I can't really say. Because I'm doing a simple binary transfer only receivers and operating systems supporting native Ir transfer working. Most of the actual IrDA receivers accept only data with IrDA protocol, they don't work in connection with native Ir transfer. So I tried to use a Pocket PC as receiver. It works fine with my Jornada 565 (distance from some up to 300 millimeters), only sometimes with my Axim X50v (only successful with a distance ~220 millimeter) and never with a HP hx2110 or a Qtech 9100. The

process was never such stable like with the 2nd approach. The only good think was, that the transfer time of the HP42S image was reduced to less than 6 min.

Because of the various problems with receivers I quit further development. There's no other common platform in the HP community like the HP48.

2. Example

This isn't an example for beginners, you must be familiar with the Pioneer integrated memory scanner and you must to know how you can get and save binary data on your receiver. Set your receiver to 2400 baud 8,N,1 please.

If you want to try, here's a "Hello world" example for the HP42S. But it should run on all High-End Pioneers, because only fixed hardware addresses are used. Type in these numbers very carefully with the memory scanner into your HP42S. A good address maybe #52000.

```
1FF0304D215D0808
F81B43418000CA13
01F0000434A20001
5A01590160170CE5
FE1ED03078108456
C6C6F60277F627C6
46000713614A1619
68A07A0060FFD281
B3FCAA0C4E434000
0481B331B18160EF
215D081CD215D031
41CE5DFFAA4E50E01
```

After starting the program with the <.> key you should exactly receive 11 bytes containing "Hello world" without the " delimiters. Got it? Congratulations you maybe able to get a ROM image by this way.

3. The ROM Upload program for 64KB ROM High-End Pioneer calculators

```
1FF0304D215D0808
F81B434A6000CA13
01F0000434A20001
5A01590160170CE5
FE1ED03034FFFF0D
5D213614A1617B00
CD53FD281B3FCAA0
C4E4340000481B33
1B18160EF215D081
CD215D03141CE5DF
FAA4E50E01
```

If it worked you should got an exact binary dump of the ROM image.

4. The ROM Upload program for the HP28S

MRIR282BB [FEE9]

```
"69C200B0008F1805
01FF0FFFD215D080
8F81B43446000CA1
301F82EFF34B2000
8F2FF401FD0FFF34
FFFF1D5D213614A1
617010CD53F80808
DB2B30FCAA0C4E43
482EFF81B331B181
60EF215D081CD215
D03141CE5DFFAA4E
```


50E010"

These are only a proof of concept programs, especially because of the big problems with the receivers. Sorry I can't and will not give any support to the published serial approach.

Christoph Giesselink
c dot giesselink at gmx dot de

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Replacing the HP-9100 decimal select switch

Posted by [David Smith](#) on 6 Feb 2006, 7:07 p.m.

The HP-9100A and HP-9100B desktop CRT calculators have a thumbwheel switch on the keyboard for selecting the number of displayed digits to the right of the decimal point. This switch is a 10 position (36 degrees per step) continuous rotary switch (with no rotation stops). It is a custom switch that has produces a unique, custom four bit code for each position.

I had a 9100A that had this switch hopelessly damaged. I repaired the machine by replacing this switch with an off-the-shelf un-encoded rotary switch and a diode matrix to produce HP-9100 codes. It turned out to be surprisingly hard to find a suitable switch. It needs to be low profile with a short 1/4" diameter shaft (3/8" threaded mounting bushing).

The switch shaft needs to be rather short. I had to cut about 1/8" off of mine. The original switch had a round shaft, but the replacment had a "D" shaped shaft. This actually makes it easier to secure the tumbwheel knob. It is held in place by two setscrews. But, it does offset the thumbwheel markings from the switch markings by two positions (your offset may vary).

I used a Grayhill part number 05001-10N-9526 rotary switch that I purchased off of Ebay for \$2.00. Digi-Key sells another Grayhill switch that looks like it would work, but it is around \$40. I used standard 1N914 or 1N4148 signal diodes.

The switch has five wires connected to it. The common wire is BLACK. The other wires are colored (in my machine): A=BLACK/WHITE (with mayby YELLOW), B=RED/WHITE, C=GREEN/BLACK/WHITE, D=RED/BLACK/WHITE,

The original switch produces the following codes (thanks to Tony Duell for confirming my original tests): 0=ABD, 1=BD, 2=AC, 3=C, 4=A, 5=(no connect), 6=ABC, 7=BC, 8=AB, 9=B

The other six possible codes produce some interesting displays on the machine, including the display of the extra guard digits used in internal calculations. When you have the old switch removed play around and see what you can get. Also verify the color coding of your wires. Curious people with a good 9100 switch can set their switch to "5" and try jumpering the various codes into the switch wiring.

I made up 10 "single inline" diode packs consisting of four diodes each with a common anode connection. The common anode connection of each diode pack was soldered to a switch contact. The diode packs were aligned so that the diode cathode leads stood straight up from the back of the switch and the packs were radially aligned towards the center of the switch. This kept all the diodes within the footprint of the switch. There is no room for them to hang over the edge of the switch.

I then snipped out the unused diodes (see the next paragraph). Next, I wired the cathode leads of the diodes into four rings. The "A" ring was the outermost ring, the "B" ring next, then the "C" ring. The "D" ring was the innermost ring. This ordering evened out the density of the connections. I formed the leads by bending them over as close to the diodes as possible.

If I was going to do this over again I would have only wired in the proper number of diodes properly positioned onto each switch pin, but by starting with uniform four diode packs it was easier to get neat, evenly spaced rings of diode leads. It is very important to minimize the profile of the diodes on the back of the switch since you have little clearance to the side of the case. If your switch plus diodes were too thick to clear the case, you could wire up the diode matrix on a small circuit board connected to the switch with 10 more wires.

After wiring all the diodes onto the switch, I attached the thumbwheel knob and then mounted the switch/knob onto the keyboard frame. It is held onto the keyboard frame by the nut on the switch bushing. Tightening the nut is a problem since there is very little clearance between the thumbwheel and keyboard frame. A very thin wrench would work, but I wound up holding the nut with needle nose pliers and turning the switch to tighten it down. Finally I soldered the common wire to the switch and the four code wires to their appropriate ring of diodes. Voila... 9100A good as new.

Edited: 6 Feb 2006, 7:17 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Repairing 9120 (and possibly other) printer platens

Posted by [David Smith](#) on 24 Jan 2006, 7:58 p.m.

The HP-9120 printer for the HP-9100A and 9100B calculators has a rubber paper advance roller (platen) . Like all things HP and rubbery the platen rubber has reverted back to its primordial goo. It is the nastiest, stickiest, black goo you can imagine.

A local surplus shop had the perfect material for repairing the platen. It is 3M Cold Shrink 8426-9 electrical connector insulator. This material is basically a rubber tube that has been stretched over a collapsable polyethylene core.

You remove the paper advance roller from the printer, measure the position of the rubber on the shaft, clean the goo off the shaft, insert the cleaned steel shaft into the core of the Cold Shrink insulator and peel out the supporting polyethylene material. The rubber insulator shrinks down onto the steel shaft. You then trim the ends of the rubber to match the size of the original roller. Voila, a REALLY nice paper advance roller. The resuting platen exactly matched the original diameter.

The Cold Shrink rubber material has a VERY grippy texture with just the right amount of "give" to it. Paper will NOT slip on the stuff.

DigiKey sells the stuff for \$16.16 A nine inch piece can do two rollers. An 11 inch piece (\$19.36) should do three rollers.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

Woodstock RAM chips

Posted by [Eric Smith](#) on 18 Jan 2006, 5:27 p.m.

There are at least five different RAM-only chips used in the Woodstock/Topcat series; some of the calculators (HP-19C, HP-27, HP-29C, HP-67, HP-92, HP-97) have some combined ROM/RAM chips which I won't describe here.

There are three different "single density" RAM chips, containing 8 registers of 56 bits each:

1. Two 1820-1564 RAM chips are found in the HP-22, HP-25, and HP-91 calculators.
2. Two 1820-1630 static RAM chips are found in some HP-25C calculators.
3. Two 1820-1843 static RAM chips are found in some HP-25C calculators.

There are two different "double density" RAM chips, containing 16 registers of 56 bits each:

1. One of the 1820-1886 static RAM chips is found in some HP-25C calculators, and three in some HP-95C calculators.
2. One of the 5061-0469 CMOS static RAM chips is found in some HP-25C calculators, two in the HP-19C and HP-29C calculators, and three in some HP-95C calculators.

These chips are all generally pin compatible. However, Viktor Toth [reports](#) that if you replace two single RAM chips with a double RAM, you have to add a 100K resistor between pins 8 and 14 of the RAM in order for continuous memory to function correctly.

The 1820-1564 chip is probably dynamic (requiring a continuous clock), so it will probably will not provide continuous memory if you try to use it in the HP-25C, HP-29C, or HP-95C.

The 5061-0469 is CMOS, while all of the 1820-xxxx RAMs listed above are probably PMOS, so the 5061-0469 probably has lower power consumption for longer battery life.

Edited: 18 Jan 2006, 5:40 p.m.

Edit Message

Delete Message

Password:

[\[Return to the Message Index \]](#)



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

Internet Protocol (v4) Addresses on the HP-16C

Posted by [Cameron Paine](#) on 20 Dec 2005, 2:36 a.m.

An IPv4 address is simply a 32-bit unsigned integer. An address is given a sub-structure by conventional rules that divide the address pool into three classes based on numerical range. The sub-structure can be further refined by applying a 32-bit mask to the address.

Conventionally, addresses and masks are written using dotted-decimal notation. In this system, each of the four octets that make up the address (or the mask) are expressed in decimal, with a dot separating each octet. The left-hand position holds the most-significant octet.

For example:

192.168.192.42 255.255.192.0

There is an alternative notation for network masks in which a single integer, representing the number of bits that are set in the mask, is appended to a dotted-decimal address, preceded by a slash character. This is sometimes called CIDR notation. Using this scheme, the above address/mask combination would be expressed as:

192.168.192.42/18

This is a useful notation because the actual mask can be conveniently generated using the number after the slash and the 16C's left mask (MASKL) operator.

The Package

The routines in this package manipulate IPv4 addresses and network masks in various useful ways. None of the functions provided are particularly ingenious but collectively they can save quite a few keystrokes.

User Routines

GSB A - JOIN dotted decimal octets to 32-bit integer
 GSB B - SPLIT 32-bit integer to dotted decimal octets
 GSB C - NETWORK address from CIDR
 GSB D - BCAST (broadcast) address from CIDR
 GSB E - Number of host ADDresses given a mask

Service Routines

GSB 9 - convert MASK in register 0 to CIDR form.
 GSB 8 - OCTET processor for SPLIT
 GSB 7 - convert mask in X to CIDR MASK in X

Input and Output

The 16C has no native dotted-decimal representation. Therefore we use the four stack registers as follows:

```
T := octet 3 (ms)
Z := octet 2
Y := octet 1
X := octet 0 (ls)
```

For those functions that require CIDR addresses, the mask should be stored in Register 0. If the mask is in binary form it will be converted to CIDR form as required. Thus 192.168.192.42/26 is entered as:

```
26 STO 0
192 ENTER
168 ENTER
192 ENTER
42
```

The four octets in a returned address can be retrieved by using the roll-down key.

All routines use I for temporary storage. They may be trivially modified to use another register for that purpose. The calculator must be in unsigned decimal mode with a 32-bit word size.

; JOIN - dotted decimal octets to 32-bit integer

```
+01 - LBL JOIN (A)
+02 - R^
+03 - STO I      ; I = 03
+04 - Rv
+05 - X-Y        ; x = 01; y = 00
+06 - [8]        ; discards T
+07 - RLn        ; x = 01 << 8
+08 - LSTx       ; bring back 8...
+09 - Rv         ; ...and put it in T
+10 - OR         ; x = 00 | 01 << 8; y = 02; z = 8; t = 8
+11 - X-Y        ; x = 02
+12 - R^         ; bring back 8...
+13 - SL         ; ...make it 16
+14 - RLn        ; x = 02 << 16
+15 - LSTx       ; bring back 16...
+16 - R^         ; bring back 8...
+17 - OR         ; ...make it 24
+18 - Rv         ; x = 02 << 16; y = 00 | 01 << 8; t = 24
+19 - OR         ; x = 02 << 16 | 00 | 01 << 8
+20 - RCL I      ; x = 03
+21 - R^         ; x = 24
+22 - RLn        ; x = 03 << 24
+23 - OR         ; x = 03 << 24 | 02 << 16 | 00 | 01 << 8
+24 - RTN
```

; SPLIT - 32-bit integer to dotted decimal octets

```
+25 - LBL SPLIT (B)
+26 - GSB OCTET ; do 00
+27 - Rv
+28 - STO I      ; stash in I
+29 - R^
+30 - GSB OCTET ; do 01
+31 - Rv
+32 - X-I        ; stash 01, stack 00
+33 - R^
+34 - GSB OCTET ; do 02 and 03
+35 - X-Y        ; x = 02; y = 03
+36 - R^
+37 - X-I        ; x = 01; y = 02; z = 03
+38 - R^         ; x = 00; y = 01; z = 02; t = 03
+39 - RTN
```

; NET - network address from CIDR host address

```
+40 - LBL NET (C)
+41 - GSB JOIN
+42 - GSB MASK
+43 - MASKL
```

```
+44 - AND
+45 - GTO SPLIT
```

```
; BCAST - broadcast address from CIDR network address
```

```
+46 - LBL BCAST (D)
+47 - GSB JOIN
+48 - GSB MASK
+49 - MASKL
+50 - NOT
+51 - OR
+52 - GTO SPLIT
```

```
; NADDR - number of host addresses on a network
```

```
+53 - LBL NADDR (E)
+54 - X=0?
+55 - RCL 0
+56 - X=0?
+57 - RTN
+58 - GSB CIDR_MASK (7)
+59 - MASKL
+60 - NOT
+61 - [1]
+62 - [-]
+63 - RTN
```

```
;
; Service Routines
;
```

```
; If the high bit of X is set we assume it contains a binary
; mask and we convert it to CIDR form. Otherwise we leave it
; alone.
```

```
+64 - LBL CIDR_MASK (7)
+65 - [3]
+66 - [1]
+67 - B?
+68 - B#
+69 - RTN
```

```
; Places the low-order octet of X in Y and the remaining
; high-order bits, right justified, in X.
```

```
+70 - LBL OCTET (8)
+71 - [8]
+72 - MASKR      ; low octet mask
+73 - X-Y        ; swap so we stash in LSTx
+74 - AND        ; clear high bits
+75 - LSTx       ; X = word; Y = octet;
+76 - [8]        ; Clear low octet so we can rotate
+77 - MASKR      ; low octet mask...
+78 - NOT        ; ...becomes remaining octet mask
+79 - AND        ; clear low octet
+80 - [8]
+81 - RRn        ; rotate cleared octet to MSB
+82 - RTN
```

```
+83 - LBL MASK (9)
+84 - RCL 0
+85 - GTO CIDR_MASK (7)
```

When entered into an otherwise empty 16C, with the word size set to 32 bits, [f][MEM] should show:

```
P - 6   r - 028
```

If you don't have a 16C handy and you'd like to try these out you can download the package for [Nonpareil here](#) or for

my [simulator here](#).

Edited: 20 Dec 2005, 2:40 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

E-mails from Everett Kaser on the HP-85

Posted by [john](#) on 19 Dec 2005, 9:29 p.m.

Hi,

I had a brief but productive exchange of e-mails with Everett in 2001/02. Since this was 3/4 years ago and have done nothing much with them, I thought I would present them here in slightly edited form (with EK's permission), as they provided an interesting view into the development of the HP-85 and EK's work on the project.

Happy Christmas! John

When Everett was working for HP he joined the development team for the HP-85 and later produced software for it including a series of games. He has many articles published in various magazines, and has many mentions on the web (see Everett Kaser links).

I first e-mailed Everett in September 2001 and listed below are extracts from his replies relating to the history of the HP 80 series.

Many thanks to Everett for kindly giving his permission to me to publish his e-mails on the WWW.

Synopsis 3rd September 2001 – introduction, HP-85 emulator 3rd September 2001 part 2 – HP magazines 18th June 2002 – introduction to HP-85 Assembler ROM, comparison HP-85, HP-71B, Kangaroo, VisiCalc port to HP-85 21st June 2002 – Everett's early career .in HP, initial Assembler, how the Assembler ROM got produced. 13th August 2002 – How the EMS ROM was produced containing two EK games Everett Kaser on the web Synopsis

Everett joined HP Corvallis in 1976 having graduated in Art. Everett joined the HP-85 team working on binary programs, and decided to work on an improved Assembler, as the previous one (for HP internal use only) was inefficient and slow. He worked on this in his spare time away from calculator production, and was called in work on it full time after product management realised that the customers for the newly launched HP-85 wanted an Assembler for it. He worked on finalising the Assembler ROM and working with a technical author on the manual for it until it was released. He then moved into the HP software applications lab.

While the HP-85 was being developed there was a battery powered laptop type computer based on the same chips being developed in parallel, the HP-75C code-named Kangaroo. The HP-71B was a development from the Kangaroo product line.

Everett left HP around 1997, for details of his current work please see <http://www.kaser.com>. He started writing an emulator for the HP-85, but this was not completed, partly due to him not having source code describing the machines tape and printer routines.

3rd September 2001 – introduction, HP-85 emulator

Subject: Re: HP 85 software query Date: Mon, 3 Sep 2001 08:06:17 -0700 From: Everett Kaser To: john Cc:

Hi, John.

jtc> I came across the FAQ for your software, which mentioned that jtc> you used to work for HP and whilst there produced several programs jtc> for the HP 85. I have one of these machines, and am very jtc> impressed with it as a landmark in computer evolution - it jtc> packs an incredible amount into a small box, and is the most jtc> accurate calculator I've ever used.

:-) I have a great fondness for the HP 85. It was the first "real" computer (other than a small, home-built microcomputer) that I ever worked on, and its architecture was quite remarkable. Unfortunately, I have very little software left from the machine, as most of what I had was stored on the tape cartridges, and they're no longer readable. Plus, when I left HP 4-5 years ago, much of my HP 85 material was left behind (or thrown out).

What little I might have is buried (literally) behind boxes and stacks of other things, and not readily accessible. As I'm VERY swamped with "staying alive work" :-)) right now, I don't want to start digging, but I WILL keep your email handy, and when the time presents itself, I'll see what I can dig up. It may be as long as 6 months, however, so please be patient.

You'll be interested to know that about a year and a half ago, I spent a week or two working on an HP 85 simulator that runs under MS Windows. I "sucked" the bytes out of the HP 85 internal ROMs and most of the external plug-in ROMs that I had (I still have a working HP 85, except for the tape driver and printer, who's rubber belts have rotted away). I then wrote a Windows program in C/C++ that emulated the HP 85 CPU, display controller and keyboard controller. The CPU was a real challenge, as it all math operations in both binary and BCD math, depending upon a flag you set (hence the accuracy of the calculator that you mentioned... most computers do the math by translating the decimal numbers we use into binary numbers, which can cause inaccuracies to creep in... many early pocket calculators would divide 2 into 4 and get 1.999999999999. :-))

Anyway, I got it barely functional before I had to set it aside and return to earning a living. :-)) I can type in a simple program and run it. It runs just about the same speed as the original HP 85. :-))) However, there are problems with the graphics display mode, and you can't print or save programs to disk. My hope is to eventually get back to it and fix the bugs in the graphics emulation, and then implement emulation of the tape drive and the printer. However, that will involve disassembly those HP 85 assembly language routines and reverse-engineering the tape and printer routines, as I no longer have any documentation on those chips. The graphics and keyboard controllers were pretty well documented in the Assembler ROM manual, but not the tape or printer controllers. The emulator that I wrote (started) includes a disassembler and the ability to add comments to the disassembly. I have source code listings to the HP 87 system ROMs, which helps a lot, as the code was a port/modification, but unfortunately the HP 87 didn't have a tape drive or printer, so those routines had been removed, and I no longer have the HP 85 system ROM listings. (You can't imagine how many times I kicked myself and beat myself about the head and shoulders for not keeping those!!! :-(

So, perhaps someday...

As I said, I'll keep your email on my "to-do" stack, and will try to get to it eventually.

Best wishes,

Everett

=====

Everett Kaser Software Sherlock, Moriarty, Honeycomb Hotel, Hero PO Box 403 Solitile, Descartes, Bumpershoot, Knarly Works Albany OR 97321-0117 Phone: 541-928-5259 8am-8pm PACIFIC TIME USA <http://www.kaser.com>

=====

3rd September 2001 part 2 – HP magazines

Subject: Re[2]: HP 85 software query Date: Mon, 3 Sep 2001 14:54:39 -0700 From: Everett Kaser To: john Cc:

Hi, John.

jtc> The FAQ mentioned you wrote for some HP magazines, could you jtc> let me know the magazine titles and periods they published jtc> HP 85 listings, I might be able to find them in a library over here.

Good luck finding them, they were definitely "small press" magazines. I believe the main one was called "HP Professional", but I'm not positive. I doubt whether I still have my copies or not, but when I get the time to "dig", that'll be one of the things I'll look for.

<cut>

Best wishes,

Everett

=====

Everett Kaser Software Sherlock, Moriarty, Honeycomb Hotel, Hero PO Box 403 Solitile, Descartes, Bumpershoot, Knarly Works Albany OR 97321-0117 Phone: 541-928-5259 8am-8pm PACIFIC TIME USA <http://www.kaser.com>

=====

18th June 2002 – introduction to HP-85 Assembler ROM, comparison HP-85, HP-71B, Kangaroo, VisiCalc port to HP-85

Subject: Re[4]: HP 85 software query Date: Tue, 18 Jun 2002 15:56:22 -0700 From: Everett Kaser To: john Cc:

Hi, John.

jtc> I hope you are well (and cheering the USA team in the World Cup!)

:-) They're doing remarkably well for themselves this year!

jtc> I now have been given a copy of the HP-85 Assembler manual, and jtc> am starting to work through it. This certainly opens up more jtc> possibilities for the machine, it is a shame that the Assembler jtc> is not a built in ROM as the machine is very limited with its fixed jtc> Basic commands, the Assembler makes it easy to extend & update [the Basic].

The Assembler ROM was an after-thought. Originally the developers hadn't even planned on offering one until after introduction and it became obvious that a LOT of customers wanted it. It was the writing of the Assembler ROM in my spare time (before the product was introduced) that got me my engineering position with the company.

jtc> I am considering getting an HP-71B as a follow on to the HP-85, jtc> is there much in common between the two please?

I don't recall. There were two projects going on at the same time, in parallel, but separate. Both used the same CPU and support chips, one was the HP-85 microcomputer, the other was a "laptop" computer, battery powered, about the size of a slim hardback book. It only had a 1-line LCD display, and all the I/O devices were different, so the language and ROM code was quite a bit different from the HP-85. I don't even remember it's product number, but its code name was Kangaroo. Later on, the HP-71B was created, which used a similar operating system as Kangaroo, but I remember even less about it's internal chip sets, not having been involved in either of those projects. But anyway, the 71B was a follow on to the Kangaroo product, not the HP-85.

jtc> Anyway, hope your company's OK, and yes I am still interested jtc> in any HP-85 software you may have, I have

only seen a handful jtc> of things on eBay which where out of my price range, and some jtc> dubious items like an untested tape with VisiCalc written in jtc> ball-point on it!

I've looked around, and I don't have any tapes or disks any longer. I appear to have left all of those behind at the factory when I left HP over 5 years ago. I do remember when VisiCalc was being ported to the HP-85 by a couple of the other software engineers. It was kind of funny, because the HP-85 has a "clear register" instruction, whereas the Intel 8080 did not. The 8080 VisiCalc code had to do a "load immediate" instruction with a value of 0. One of the engineers (the smarter one, in my opinion) wanted to convert those loads-of-zero instructions into clear-register instructions (fewer bytes, do the same thing), but the other engineer was adamant that they also do load-zero instructions, making NO modifications to the original "logic" at all. :-)

Anyway, sorry I can't help with any HP-85 software. That was a LONG time ago. I wish I still had all of the old games that I wrote for the HP-85! Sigh.

Best wishes,

Everett

=====

Everett Kaser Software Sherlock, Moriarty, Honeycomb Hotel, Hero PO Box 403 Solitile, Descartes, Bumpershoot, Knarly Works Albany OR 97321-0117 Phone: 541-928-5259 8am-8pm PACIFIC TIME USA <http://www.kaser.com>

=====

21st June 2002 – Everett's early career in HP, initial Assembler, how the Assembler ROM got produced.

Subject: Re[6]: HP 85 software query Date: Fri, 21 Jun 2002 20:58:59 -0700 From: Everett Kaser To: john Cc:

Hi, John.

jtc> You wrote that you started to write the Assembler ROM before jtc> you gained an engineering position, so what area were you jtc> working in before please?

Short version of a long story:

I started college in Electrical Engineering in 1971. Decided (after 1 1/3 years) that it wasn't right for me and switched to Art with a specialization in Custom Designed Jewelry & Metalsmithing (graduated with that degree). Decided I didn't really want to work in jewelry, so got a hired onto HP in Corvallis a month after the plant opened in 1976, as a "Production I" employee (making a whole \$580/month...NOT take-home, but gross). Over the next 2-3 years I worked my way up to "Production II" and then "Technician I" and then "Technician II" (this was in the calculator production area).

In college I'd gotten interested in John Conway's "Game of Life" (cellular automata), and wanting to do that in some way other than manually on graph paper got me interested again in electronics, and microcomputer kits were just becoming available then. I built an RCA Cosmac 1802 Elf (256 BYTES of RAM) computer, and learned to program it in machine code (no assembler even :-), just toggle switches and an input pushbutton).

When the HP-85 was getting close to release (about 9 months to a year before entering production) I found out about it and asked the other production tech (who was assigned to it) to take me into the lab and introduce me to the lab engineers, as I wanted to learn to program it (to do the Game of Life, of course). One thing lead to another, and I ended up become quite good at writing binary programs for it. But, the only assembler ran on an HP 1000 minicomputer. One of the engineers had written a BASIC assembler (with a binary program that would store the output to tape as a "binary program" type of file), but it was HORRENDOUSLY slow and ugly to use. (Still with me? :-)

One of the applications engineers had asked me if I'd like to try converting his Fast Fourier Transform subroutine from BASIC into assembly to speed it up. I didn't remember anything about FFT's from my engineering days, but decided it was just a "translation" thing, so agreed to try. When I was done, it was a 750 line assembly language program that took 30 minutes to assemble with the BASIC assembler. So, I decided I was going to write an assembly language assembler as my next project. I was going to do it as a binary program, loaded off of tape, but the lab engineers convinced me to do it as a ROM (we had special boards that we could plug EPROMS into), so I agreed. I was doing all of this programming early in the morning before my normal production job started, and late in the afternoon, and in the evenings at home.

When the HP-85 was introduced in January 1980, all of the lab/marketing engineers and managers went out on New Product Tour (sales offices, retail stores, etc) to introduce it. When they came back, they realized that everyone wanted to know if there was an assembler available for it (assembly language was BIG back then!) They said, "Sh*t, maybe we'd better get an assembler product written for this thing!!!" The lab engineers cleared their throats and said, "Hrrmmm. We just happen to know someone who has one almost done..." (It was functional by that point, but not polished and completely finished.) All of a sudden I found myself demoing my assembler ROM to all of the managers from first-level clear up to the lab manager and the marketing manager (one-step down from the division manager). Scary stuff for a lowly production tech. :-)

Anyway, they talked to me, to my production supervisor, and all of a sudden I find myself on loan to the lab to finish it up and work with a contract writer on the manual. When it was done and ready for release, they offered me a job as a software applications engineer (August 1980). Eventually, I moved into the lab and worked there from then on.

Ok, I guess that wasn't so short. It's just a tough story to tell in one paragraph... :-))

Best wishes,

Everett

=====

Everett Kaser Software Sherlock, Moriarty, Honeycomb Hotel, Hero PO Box 403 Solitile, Descartes, Bumpershoot, Knarly Works Albany OR 97321-0117 Phone: 541-928-5259 8am-8pm PACIFIC TIME USA <http://www.kaser.com>

=====

13th August 2002 – EMS ROM and its EK games

Hi, John.

J> Although I have an HP-85 I also have the HP-87 EMS ROM (long story..). I J> looked through the contents of the ROM (using Assembler ROM & a PC) and J> there seems to be a game in there. Here are the strings I have found:

I was working in the R&D lab at the time, and a good friend was working in marketing. This company (Structured Software Systems, I imagine, tho I don't remember the name for sure) approached HP about getting a custom ROM made with their own code in it. HP was happy to (pay the price, get whatever you ask for). Well, I was delegated the job of taking their bits and making the release tape to send off to the ROM factory. I noticed, though, that the code they wanted in the ROM only took up maybe 1/2 of the available 8K bytes space. And I had a few little games around that I'd written in my spare time... I thought, "What the heck? What they don't know won't hurt anyone, right?" (I was young and foolish... :-) So, I added the binaries of a couple of the games and gave them ridiculous statement names to invoke them (so that they wouldn't ever collide with anything else in the name space). All was well until the ROMs came back and were sent to the purchaser. One day my friend from marketing wanders into the lab to my desk and asks, "Say, Everett.... you didn't happen to put anything extra into those ROMs did you?"

Well, with a sheepish grin, I had to admit what I'd done. It turns out that the ROM purchasers had apparently written some binary programs that, rather than calling the code in the ROMs through the normal vector tables, jumped

DIRECTLY into the ROMs to a specific address... and adding the two new statement names to the name tables and their corresponding addresses to the address tables had (because those tables are always at the top of the ROM's address space) pushed everything else down a few bytes.

Oops.

Oh well, all they had to do was recompile their binaries with the new addresses and all was well. Fortunately they didn't complain (apparently), as I never heard any more about it.

And that's the story.

Best wishes,

Everett

=====

Everett Kaser Software Logic & Puzzle Games for PCs PO Box 403 Albany OR 97321-0117 Phone: 541-928-5259
8am-8pm PACIFIC TIME USA <http://www.kaser.com>

=====

J> I find it hard to believe that the straight-laced HP corp would J> permit some games code in the ROM - was this public knowledge J> before?

:-) No, HP would NOT have given permission had I asked. Forgiveness is easier to get... :-) So far as I know, the story has never been told publicly. Feel free.

Best wishes,

Everett

=====

Everett Kaser Software Logic & Puzzle Games for PCs PO Box 403 Albany OR 97321-0117 Phone: 541-928-5259
8am-8pm PACIFIC TIME USA <http://www.kaser.com>

=====

Everett Kaser on the web

There are several thousand hits on a search for “Everett Kaser”, thanks to his work on producing a series of highly respected puzzle games. Here a few of the places on the web where Everett’s work with HP crops up.

Everett’s homepage <http://www.kaser.com> has a FAQ that mentions the games he wrote for HP series 80.

The Evolution of HP Palmtops – an interview with Everett: <http://www.palmtoppaper.com/ptphtml/12/pt120055.htm>

Windows CE review – including picture and a precis of Everett: <http://www.pocketpcmag.com/Premier/kaser.htm>

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

"Double Memory" for the HP-33s

Posted by [Paul Brogger](#) on 12 Dec 2005, 6:41 p.m.

This is a program to store up to 58 half-precision values as encoded "half-registers" in HP-33s variables.

To store a value, put the value itself in stack y, and the storage location ($1 \leq x \leq 58$) in stack x, and "XEQ D". The routine stores the value, and returns the value as stored (normalized to 5 digits) in stack x.

To recall a value, put the negated storage location in stack x ($-1 \geq x \geq -58$) and "XEQ D". The routine returns the specified value in stack x.

After a "store" call, stack contents have been lost; for recall, stack y is returned in place. As written, the routine uses flags 3 & 4, but it could probably use other, more obscure flags & leave 3 & 4 to the calling programs.

The program starts mapping pseudo-registers to actual storage way up in the 6 statistics regs (half-register locations 1-12), and proceeds down to register A (half-registers 57 & 58). So if the calling routine needs a few regs, it can keep its storage low in the alphabet, and limit the range of pseudo-index values employed.

The program utilizes storage registers (variables) X, Y & Z for intermediate values, and restores the original contents of i before returning.

Potentially, similar techniques could be used to store reals of greater precision but reduced range (6 digits & 10^{-9} -- 10^{+9}), 7-digit integers, or even more, smaller integers (perhaps as many as 87 4-digit values?)

Breaking it apart to utilize several labels would make it shorter overall, faster, and easier to understand & maintain. But I was going for the extremes.

There *may* be problems possible when the routine is first run against uninitialized storage registers. But once it has successfully initialized the storage areas to recognized values (say, zero), it has solidly endured literally hours of battery-draining random-value testing. (You may wish to run a loop that pre-sets all half-registers to be used, before starting your "production" routine.)

Half-register encoding:

Two values ("half-registers") assumed to be in scientific notation with the forms:

half-register 0
 $\pm a.bcde \times 10^{\pm fg}$

half-register 1
 $\pm A.BCDE \times 10^{\pm FG}$

are combined into each hybrid register value with the form:

hybrid register
 $\pm a.bcdeABCDEF \times 10^{\pm Sfg}$

where the digit "S" encodes the signs of half-register 1's mantissa and exponent:

Signs in half-register 1 values	value of "S"
+A.BCDE*10 ⁺ FG	0
-A.BCDE*10 ⁺ FG	1
+A.BCDE*10 ⁻ FG	2
-A.BCDE*10 ⁻ FG	3

Half-registers have five digits of precision, and may assume any positive or negative value with magnitude in the range
1.0000 * 10⁻⁹⁹ <= n <= 9.9999 * 10⁹⁹
(Exception: To insure the encoding works consistently, the value -1.0000*10⁻⁹⁹ is used to encode zero.)

Flags 3 & 4 are used to indicate various conditions throughout processing, and at least of of them will be set upon return.

Register (variable) Usage

i index	name	user index	Use(s)
01 - 23	A - W	58 - 13	storage - two half-register values in each
24	X	- -	half-register 0 (-9.9999*10 ⁹⁹ <= n<= 9.9999*10 ⁹⁹)
25	Y	- -	half-register 1 (-9.9999*10 ⁹⁹ <= n<= 9.9999*10 ⁹⁹)
26	Z	- -	storage location and exponents work area
27	i	- -	index register (for indirect addressing of other regs)
28 - 33	stat regs	12-01	storage - two half-register values in each

Program Listing

Line #	LBL	OP	Comments
D0001	LBL	D	"DblMem" - HP-33s "Double Memory" subroutine
D0002	CF	4	Normalize and Store Caller's Index Specification
D0003	x < 0?		if set, exit after extracting half-register 1
D0004	SF	4	if x negative, "recall" requested
D0005	ABS		so indicate such with flag # 4
D0006	IP		now that sign has been encoded, get rid of it
D0007	1		and insure only integer value
D0008	x > y?		minimum value for pseudo-index
D0009	x <> y		if caller specified a lower value
D0010	Rdn		use 1.
D0011	58		maximum value for pseudo-index
D0012	x < y?		if caller specified a higher value
D0013	x <> y		use 58.
D0014	Rdn		
D0015	68		Map index specified
D0016	x <> y		to storage available
D0017	-		by first "reversing" it and adding 9.
D0018	CF	3	If set, pseudo-index will not require adjustment
D0019	56		pseudo-index of 1st stats register
D0020	x <= y?		If caller specified that or greater
D0021	SF	3	then signal no later adjustment necessary
D0022	Rdn		
D0023	8		Pseudo-index gap for "system" registers X, Y, Z & i
D0024	FS?	3	Is user's index at or beyond 1st statistics register?
D0025	CL x		then no adjustment necessary.
D0026	-		Shift lower storage values below X.
D0027	100		Shift exponent two digits
D0028	÷		to the right of the decimal point

D0029 ST0 Z and store pseudo-index (as fractional portion) in work area.

D0030 CF 3 Encode "read 0" vs. "read 1" as Flag 3
 D0031 FS? 4 if set, exit after extracting half-register 1
 D0032 SF 3 is this to be a read operation?
 D0033 0.02 if so, we might want to exit early
 D0034 RMDR Divide by two and multiply by 100
 D0035 x > 0? so |x| mod 2 ("even or odd") indicates which desired
 D0036 CF 3 x non-zero? (i.e., "odd")
 D0037 Rdn if so, exit only after extracting half-register 1.

D0038 RCL Z Obtain Hybrid Value from Indicated Register
 D0039 0.02 obtain caller's pseudo-index value again
 D0040 INT ÷ divide by 2 (& multiply by 100)
 D0041 x <> i translating to actual index number
 D0042 RCL (i) swap with calculator's index register.
 D0043 x <> y obtain the indicated hybrid storage value
 D0044 ST0 i restore original i value
 D0045 Rdn to minimize overall impact
 and discard

D0046 ENTER Extract Half-Register 0 from Hybrid in x
 D0047 ABS copy to y for later
 D0048 x ^= 0? insure that it's positive
 D0049 LOG x non-zero?
 D0050 INTG obtain LOG so we can get at
 D0051 SGN the exponent (characteristic)
 D0052 x ^= 0 ? -1, 0 or 1 for adjustment
 D0053 ST0 * Z If value won't wipe out the fractional portion,
 D0054 Rdn incorporate it in the exponent work area
 D0055 LAST x full characteristic (sign + magnitude)
 D0056 ST0 + Z can now be stored, maintaining fractional portion.
 D0057 10^x raise as power of ten
 D0058 ÷ and remove from original hybrid value
 D0059 E4 Shift resulting mantissa
 D0060 * four digits to the left.
 D0061 IP Get rid of half-register 1's digits
 D0062 E4 and shift four digits
 D0063 ÷ back to the right.
 D0064 RCL Z Obtain normalized hybrid exponent and pseudo-index
 D0065 IP Get rid of pseudo-index (fractional portion)
 D0066 E2 Shift exponent value
 D0067 ÷ two digits to the right.
 D0068 FP Get rid of half-register 1's signs
 D0069 E2 and shift two digits
 D0070 * back to the left.
 D0071 10^x Raise as power of ten
 D0072 * and finish extraction of half-register 0's value.

D0073 -1E-99 Substitute Special Value if Zero
 D0074 x <> y DbLMem's special encoding of "zero"
 D0075 x = 0 ? swap with recovered HR0 value
 D0076 x <> y Recovered value equal to zero?
 D0077 ST0 X then store encoded value instead.
 D0078 x <> y Store extracted half-register 0 value in X
 D0079 Rdn and clear comparand
 from stack.

D0080 -1E-99 Handle Encoded Zero, Returning Zero Itself
 D0081 x <> y DbLMem's encoding of value "zero"
 D0082 x = y ? swap with stored HR0 value
 D0083 CL x Stored HR0 value equal to encoded zero?
 D0084 x <> y Then return actual zero value
 D0085 Rdn and clear comparand
 from stack.

D0086 FS? 3 Conditional Return: Half-Register 0 Read
 D0087 RTN If the caller wanted half-register 0,
 return

Continue, Obtaining Half-Register 1

D0088	Rdn	Clear stack of half-register 0 value
D0089	RCL Z	Obtain Hybrid Value from Indicated Register
D0090	ABS	obtain caller's pseudo-index value again
D0091	FP	Remove exponent's sign
D0092	0.02	and its digits, leaving fractional index value
D0093	INT ÷	Divide by 2 (& multiply by 100)
D0094	x <> i	translating to actual index number
D0095	RCL (i)	swap with calculator's index register.
D0096	x <> y	obtain the indicated hybrid storage value
D0097	STO i	restore original i value
D0098	Rdn	to minimize overall impact and discard
D0099	ABS	Extract Half-Register 1 from Hybrid in x
D0100	RCL Z	Don't need hybrid's sign for HR1
D0101	IP	First, obtain hybrid's exponent.
D0102	10^x	reduce to exponent itself;
D0103	÷	Raise as power of ten
D0104	RCL Z	and remove from hybrid value.
D0105	ABS	Obtain hybrid exponent again
D0106	FP	don't need sign of hybrid's exponent anymore
D0107	STO Z	nor its digits
D0108	Rdn	Save pseudo-index value for later
D0109	LAST x	and discard
D0110	IP	full characteristic (sign + magnitude)
D0111	E2	(stripped of fractional index)
D0112	÷	shift two places
D0113	IP	to the right, and
D0114	2	reduce to HR1's encoded signs.
D0115	CF 3	HR1's sign-of-exponent encoded in hundreds digit.
D0116	x <= y ?	Use flag 3 now to indicate sign of HR1's exponent.
D0117	SF 3	If y >= 2, then
D0118	x > y ?	sign of HR1 exponent is negative.
D0119	CL x	If y < 2, then
D0120	-	don't subtract out HR1's sign of exponent.
D0121	SGN	Reduce to HR1 sign of value.
D0122	+ / -	Insure value is either 1 or 0.
D0123	x = 0?	Negate if anything left.
D0124	10^x	If HR1 is to be positive
D0125	STO * Z	create a positive 1, and
D0126	Rdn	store HR1 sign in exponent work area.
D0127	E4	Discard encoded HR1 sign.
D0128	*	Shift hybrid mantissa four digits
D0129	FP	to the left, and.
D0130	E5	discard HR0's digits.
D0131	*	Shift HR1's digits 5 places
D0132	IP	to the left, and
D0133	LAST x	obtain HR1's mantissa
D0134	FP	Restore all HR1 digits
D0135	E2	and obtain exponent.
D0136	*	Shift exponent two places
D0137	IP	to the left,
D0138	FS 3?	(insure no fractional portion) and
D0139	+ / -	(BTW, if flag 3 was set above,
D0140	10^x	then HR1 exponent is negative)
D0141	x <> y	raise as power of ten.
D0142	E4	Swap exponent and mantissa,
D0143	÷	shift mantissa four digits
D0144	*	to the right, and
D0145	RCL Z	combine HR1 mantissa and exponent.
D0146	SGN	and finally,
D0147	*	reimpose HR1's sign.
D0148	-1E-99	Substitute Special Value if Zero
D0149	x <> y	DblMem's encoding of value "zero"
D0150	x = 0 ?	swap with recovered HR1 value
D0151	x <> y	Recovered value equal to zero?
D0152	STO Y	then store encoded value instead.
D0153	x <> y	Store extracted half-register 1 value in Y
D0154	Rdn	discard comparand
D0155	-1E-99	Handle Encoded Zero, Returning Actual Zero
		DblMem's encoding of value "zero"

D0156	x <> y	swap with actual HR1 value
D0157	x = y ?	Actual value equal to encoded zero?
D0158	CL x	Then make it so.
D0159	x <> y	discard
D0160	Rdn	comparand
D0161	CF 3	Conditional Return: Half-Register 1 Read
D0162	FS? 4	(clean up the other flag we used)
D0163	RTN	If the caller wanted half-register 1, return
D0164	Rdn	Continue, Storing New Half-Register Value Supplied as y Clear stack of HR1 value, moving original y to x
D0165	-1E-99	Normalize Caller's y Value for Storage
D0166	x <> y	Encoded value for zero.
D0167	x = 0 ?	Swap with actual value
D0168	x <> y	and if that value is indeed zero,
D0169	x <> y	then use our encoded substitute.
D0170	Rdn	Clean up - keeping the stored, normalized value and discard the other, unstored alternative.
D0171	x < 0 ?	Encode sign of stack y value
D0172	SF 3	in flag # 3.
D0173	ABS	Ignore the supplied value's sign, for now.
D0174	SCI 4	Set display to precision we'll be supporting
D0175	RND	and enforce that precision on user's value.
D0176	9.9999E99	Upper bound of supported range
D0177	x < y ?	If supplied value is beyond that
D0178	x <> y	then substitute upper bound
D0179	Rdn	
D0180	+1E-99	Lower bound of supported range.
D0181	x > y ?	If supplied value is less than that
D0182	x <> y	then substitute lower bound
D0183	Rdn	
D0184	FS? 3	Get original stack y sign
D0185	+ /-	and reimpose on our normalized value.
D0186	RCL Z	Overwrite Appropriate Restored HR with New y Value
D0187	ABS	obtain caller's pseudo-index value / exponent again
D0188	FP	we can ignore the exponent's sign
D0189	STO Z	and fractional portion is what we're interested in
D0190	0.02	keep caller's pseudo-index value only, for now
D0191	RMDR	divide by 2 (& multiply by 100)
D0192	SGN	Reduce to an "even or odd" flag
D0193	24	indicated by zero or one
D0194	+	and calculate
D0195	x <> i	the actual index value (to variable X or Y).
D0196	x <> y	swap with calculator's index register.
D0197	STO (i)	set aside original i value
D0198	Rdn	update X or Y, as indicated
D0199	STO i	and discard
D0200	Rdn	restore original i value and discard
D0201	RCL Y	Synthesize New HR0-HR1 Hybrid for Storage
D0202	CF 3	HR1's value
D0203	x < 0 ?	Clear "HR1 sign is negative" flag
D0204	SF 3	if HR1 is less than 0, then
D0205	ABS	HR1's sign is negative.
D0206	x ^= 0 ?	insure that it's positive
D0207	LOG	if HR1 is non-zero, then
D0208	INTG	obtain LOG so we can get at
D0209	SGN	the characteristic
D0210	x ^= 0 ?	(first, we'll need its sign)
D0211	STO * Z	if HR1 is non-zero, then
D0212	Rdn	store HR1's sign of exponent on fractional index
D0213	LAST x	and discard
D0214	STO + Z	so that now, full characteristic
D0215	CF 4	may be added without corrupting fractional index.
D0216	x < 0 ?	Clear "HR1 exponent is negative" flag
D0217	SF 4	if HR1 exp is less than 0, then
D0218	Rdn	HR1's exp is negative.

D0219	RCL	Y	Obtain HR1 value
D0220	ABS		eliminate sign (encoded above as flag 3).
D0221	RCL	Z	Obtain HR1's
D0222	IP		exponent value
D0223	10^x		Raise as power of ten
D0224	÷		and remove from HR1's original value
D0225	RCL	Z	Obtain exponent / index storage value
D0226	ABS		Remove exponent's sign
D0227	FP		and its digits.
D0228	STO	Z	Update work register for now
D0229	Rdn		and discard
D0230	LAST	x	Restore full work register value
D0231	IP		and consider the exponent
D0232	E6		Shift six places
D0233	÷		to the right
D0234	+		and tack onto the end of HR1's mantissa.
D0235	STO	Y	Place HR1's encoded value in its original storage
D0236	Rdn		and discard (for now).
D0237	RCL	X	Read the HR0 value
D0238	ABS		insure that it's positive
D0239	x ^= 0 ?		If HR0 is non-zero, then
D0240	LOG		obtain LOG so we can get at
D0241	INTG		the exponent (characteristic).
D0242	SGN		-1, 0 or 1 for adjustment value
D0243	x ^= 0 ?		If resulting value won't obliterate fractional index
D0244	STO * Z		go ahead and save it.
D0245	Rdn		
D0246	LAST	x	Full characteristic again
D0247	STO + Z		store as integer portion of work register.
D0248	5		Shift that exponent five digits
D0249	-		to the right,
D0250	10^x		Raise as a power of ten, and
D0251	RCL * Y		incorporate in HR1's encoding
D0252	RCL	X	Get HR0's value
D0253	SGN		to utilize its sign
D0254	x = 0 ?		if HR0 is zero, then
D0255	10^x		make it one
D0256	*		incorporate HR0's sign in HR1's digit encoding.
D0257	STO + X		& append HR1's encoded digits to HR0
D0258	Rdn		
D0259	E2		HR1's signs are encoded in the hybrid exponent's 100's digit
D0260	RCL	Z	Obtain HR0's exponent
D0261	IP		to maintain
D0262	SGN		its sign
D0263	x = 0 ?		(if HR0 exponent is zero, then
D0264	10^x		make it "positive")
D0265	*		in the upcoming encoding operations.
D0266	10^x		Raise as power of ten
D0267	FS 3?		If HR1's sign is negative,
D0268	STO * X		add ±100 to the hybrid's exponent.
D0269	x²		Double the exponent code value
D0270	FS 4?		If HR1's exponent sign is negative,
D0271	STO * X		add ±200 to the hybrid's exponent.
D0272	Rdn		
D0273	RCL	X	Recall the newly-synthesized hybrid value.
Store Updated Hybrid in x Into Appropriate Register			
D0274	RCL	Z	obtain caller's half-register index again
D0275	ABS		we can ignore the exponent's sign
D0276	FP		and fractional portion is what we're interested in
D0277	0.02		divide by 2 (& multiply by 100)
D0278	INT ÷		translating to actual index number
D0279	x <> i		swap with calculator's index register.
D0280	x <> y		set aside original i value
D0281	STO (i)		update storage register with new hybrid value
D0282	Rdn		
D0283	STO i		discard
D0284	Rdn		original i value

D0285	RCL	Z	Initiate Recall of Newly-Stored Value
D0286	ABS		Obtain pseudo-register index
D0287	FP		without exponent's sign
D0288	E2		or digits.
D0289	*		Shift two digits
D0290	56		to the left,
D0291	CF	3	and compare to lowest index for stat registers
D0292	x <= y ?		Initialize "adjustment needed" flag
D0293	SF	3	Pseudo-index points to stat regs?
D0294	Rdn		then no need to make an alteration.
D0295	8		discard comparand.
D0296	FS?	3	Adjustment value - bypass X, Y, Z & i
D0297	CL x		Is adjustment necessary?
D0298	+		no.
D0299	68		Undo adjustment (if any) performed at entry.
D0300	-		Pseudo-register reversal
			gives "recall" value as if supplied by user.
D0301	FIX	2	Restore (my) preferred display mode
D0302	GTO	D	and go read specified value.

MEM usage: 1338

Checksum: CA99

If you've any questions or corrections, try the Forum, or take the "NoHormel" out of my email address & get to me.
Enjoy!

-- Paul B. 12/12/2005

Edited: 2 June 2006, 10:09 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

HP41C MCODE MOVE

Posted by [Doug Wilder](#) on 5 Nov 2005, 2:19 a.m.

A More Usefull MOVE

C = bbbbeeeedddd begining ending destination

The word at bbbb is always moved to dddd, all fields are arbitrary. One can grab the bottom word and move the block by pulling the bottom to dddd. That is, bbbb > eeee or eeee > bbbb is allowed. Keeping it general requires considering four cases. Laying them out with higher address on top (arrows indicate first move required):

1	2	3	4
b>e + d>b	b>e + b>d	e>b + d>b	e>b + b>d
->dddd xxxx xxxx bbbb-> xxxx xxxx xxxx xxxx aaaa xxxx xxxx eeee	bbbb xxxx xxxx xxxx dddd xxxx-> xxxx xxxx xxxx xxxx xxxx ->aaaa	->aaaa xxxx xxxx eeee-> xxxx xxxx xxxx xxxx dddd xxxx bbbb	eeee xxxx xxxx xxxx aaaa xxxx-> xxxx xxxx xxxx xxxx xxxx ->dddd

Notice that an upward move requires a decrementing counter, and a downward move requires an incrementing counter.

C = bbbbeeeedddd begining ending destination

92C2	0E	00E	A=0	! initialize and test
92C3	1E	02E	B=0	
92C4	70	0E0	SLCTQ	
92C5	AS	15C	PT= 6	
92C6	50	0A0	SLCTP	
92C7	0S	01C	PT= 3	
92C8	DS	1BC	RCR 11	
92C9	5I	0B2	A<>C PQ	dddd -> A(M)
92CA	PS	33C	RCR 1	
92CB	N6	2E6	?C#0 X	op input too many digits ?
92CC	37	067	JC +0C	
92CD	LS	2BC	RCR 7	
92CE	7I	0F2	B<>C PQ	bbbb -> B(M)
92CF	7S	0FC	RCR 10	
92D0	A0	158	STOM	eeee -> M(M)
92D1	G0	21A	RC+A M	C = dddd+eeee
92D2	5E	0AE	A<>C	
92D3	C0	19A	A=A-B M	
92D4	17	027	JC +04	trying to write below 0000 ?
92D5	5E	0AE	A<>C	C = aaaa = dddd+eeee-bbbb

92D6	KS	29C	PT=	7	
92D7	J2	262	DECC	PT	trying to write above FFFF ?
92D8	5L	0B5			
92D9	52	0A2	GONC	282D	"ERRDE"
92DA	H2	222	INCC	PT	
92DB	E0	1D8	C<>M		
92DC	3E	06E	A<>B		

at this point	for both upmove and dnmove entry:
C(M) = eeee	C(M) = ssss = source (eeee or bbbb)
B(M) = dddd	B(M) = rrrr = receiver (dddd or aaaa)
A(M) = bbbb	A(M) = tttt = terminis (bbbb or eeee)
M(M) = aaaa	

	92DD	0Q	31A	?A<C	M	?e>b
	92DE	1F	02F	JC	+05	"e>b"
b>e	92DF	PQ	33A	?A<B	M	?d>b
	92E0	2R	05B	JNC	+0B	"b>e+b>d"
1 b>e+d>b	92E1	5E	0AE	A<>C		
	92E2	5J	0B3	JNC	+16	"upmove"
e>b	92E3	PQ	33A	?A<B	M	?d>b
	92E4	1B	02B	JNC	+05	"e>b+b>d"
3 e>b+d>b	92E5	E0	1D8	C<>M		
	92E6	7E	0EE	B<>C		
	92E7	E0	1D8	C<>M		
	92E8	43	083	JNC	+10	"upmove"
4 e>b+b>d	92E9	5E	0AE	A<>C		
	92EA	13	023	JNC	+04	"dnmove"
2 b>e+b>d	92EB	E0	1D8	C<>M		
	92EC	7E	0EE	B<>C		
	92ED	E0	1D8	C<>M		
dnmove	92EE	PG	330	FETCH		<-
	92EF	7Q	0FA	B<>C	M	
	92F0	20	040	WROM		
	92F1	HQ	23A	INCC	M	
	92F2	7Q	0FA	B<>C	M	
	92F3	HQ	23A	INCC	M	
	92F4	0Q	31A	?A<C	M	
	92F5	UB	3CB	JNC	-07	->
	92F6	V0	3E0	RTN		
	92F7	JQ	27A	DECC	M	<-
upmove	92F8	PG	330	FETCH		
	92F9	7Q	0FA	B<>C	M	
	92FA	20	040	WROM		
	92FB	JQ	27A	DECC	M	
	92FC	7Q	0FA	B<>C	M	
	92FD	RQ	37A	?A#C	M	
	92FE	UF	3CF	JC	-07	->
	92FF	V0	3E0	RTN		

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

GTO

[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

My 2225B Thinkjet printer battery pack is dead: How can I recell it ?

Posted by [Etienne Victoria](#) on 25 Oct 2005, 5:10 p.m.

Credits: Information in this article is sourced mainly from Tony Duell's and Katie Wassermann former posts in the MoHpc forum along with my own experiments.

To rebuild the 2225B battery, you will need : 6 Sub Cs Ni-CADs or NiMh, one picofuse if the one in the pack is burned, strength.

Choosing Ni-CADs or NiMh is a matter of use and taste. You'll get a higher capacity with NiMh but a higher self discharge rate. NiCads provide the reverse strongpoints and drawbacks.

DO NOT mix both types. If you're not sure of yourself about the charging mode, I advise you to use NiCads.

a) Open the thing : The case is not welded but snapped, just exert enough pressure with the largest flat screwdriver as the innards are tuck with very strong double-sided tape : after releasing the cover on all edges (it's got clips all round), use a flat blade to prise it from the cells.

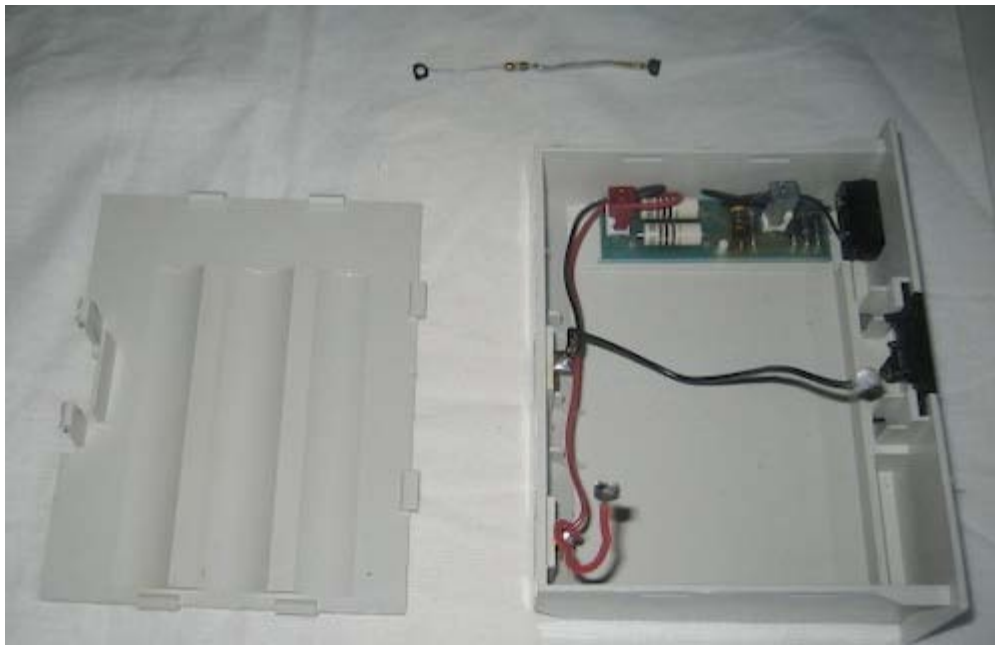
NOTE polarity of the existing cells connection alongwith picofuse position!!!

The picofuse is a 25V 5A fuse as pictured:



b) Desolder and dispose/recycle the old cells SAVING the picofuse you'll find between them. Final empty shell looks like this:

My 2225B Thinkjet printer battery pack is dead: How can I recell it ?



- c) Assemble the new cells in series with the picofuse (picofuse position & polarity not strategic)
- d) Resolder to the pack circuit, respecting polarity (+ to red wire, - to black wire) to wires shown:



and insert in pack with very strong double-sided tape **ON THE BOTTOM**. This will avoid the cells moving randomly in the pack except if you want to build special marracas.

e) If you really wish to recreate new problems and stress when opening again, apply same tape to secure the upper cover.... Otherwise refrain...

f) Reassemble cover.

g) For the first charge, charge the pack **OUTSIDE** the Thinkjet, monitoring carefully smells, gassing, funny noise, sparkles....

My 2225B Thinkjet printer battery pack is dead: How can I recell it ?

g) Test voltage at 7,2V loaded and use.

Edited: 23 Feb 2009, 4:49 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

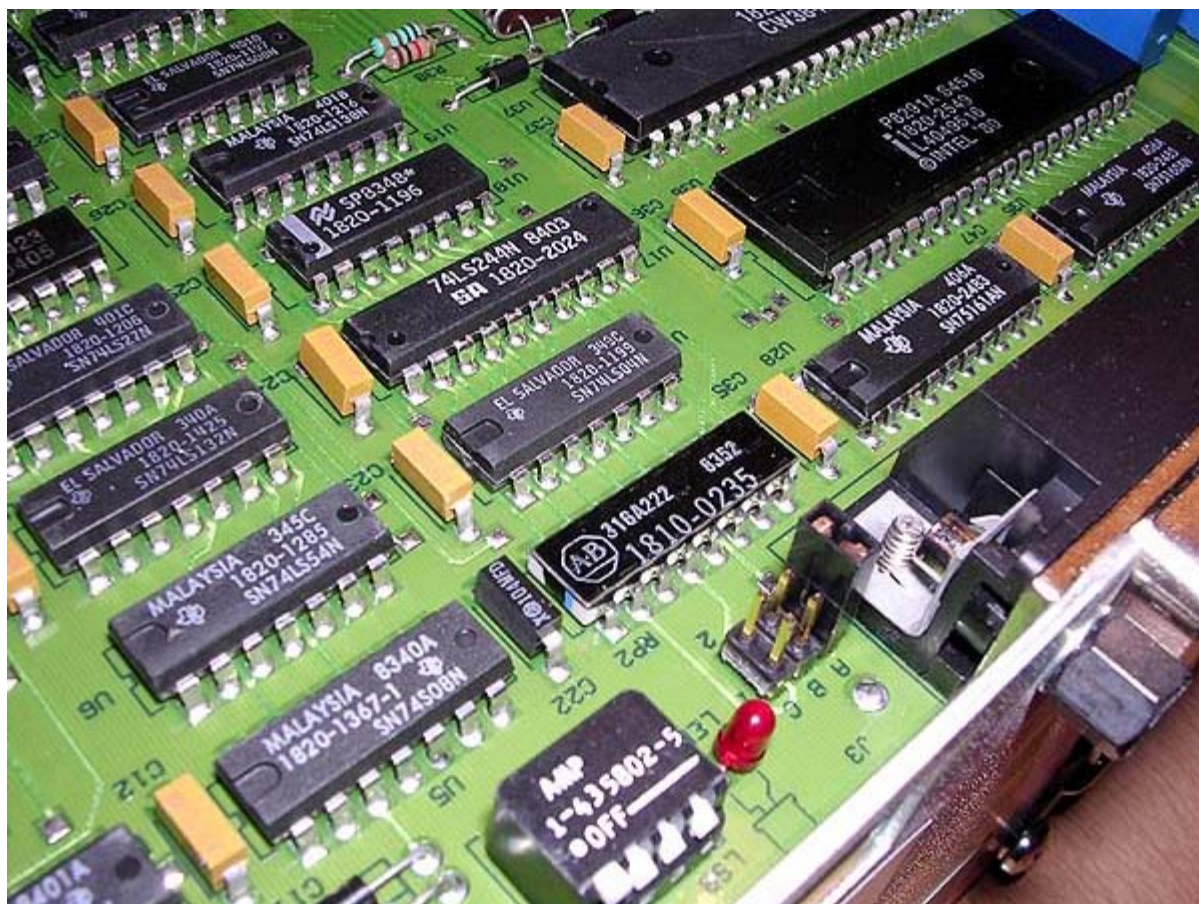
Accessing an HP-9133XV disk drive with an HP-85

Posted by [Juergen \(CH\)](#) on 12 Oct 2005, 8:56 a.m.

A few months ago, I aquired a new-old-stock HP-9133XV disk drive (3.5" single-sided floppy plus 15 MB hard drive, Amigo protocol) for CHF 20.- (about \$15.-). I thought I could use it with my beloved HP-85A. But no luck! I tried again and again, but I was not able to access the floppy drive nor the hard disk drive: the INITIALIZE command never finished. The manual states that only Option 004 of this device is compatible with Series 80 computers. Option 004 means that the logical configuration of the hard disk drive is only 5 MB with 4 volumes. But that has nothing to do with the floppy drive! Anyway, I found a hint on the back of the device that somewhere there must be three jumpers which define the logical configuration.



I opened the 9133XV, found those jumpers and configured it appropriately. But still no luck! I was at the end of my tether...



Last week, I started another try (sometimes I'm really strongheaded). I also dug out my HP-86; this was my last chance to succeed. And Yippee! The HP-86 did format floppies and the four hard disks without any problem. Now, I connected the 9133XV to my HP-85 and turned on both. A moment full of suspense! Wow, the HP-85 was able to access the floppy drive and the four hard disk volumes. LOAD, STORE, everything works fine now. I still cannot format floppies on my HP-85 but that's only a small inconvenience.

Fortunately, I also found a PC floppy drive which seems to be capable of writing to SD floppy disk using HP's LIF Utilities for PC. Now, I can transfer files between my PC and my HP-85 :-)

I would like to thank Vassilis Prevelakis and John Shadbolt for their excellent web sites with very interesting information about Series 80 computers. And many thanks to Etienne Victoria whose support last week motivated me to peg away.



Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Transferring files from Series 80 Emulator to HP-85 via floppy disks

Posted by [Juergen \(CH\)](#) on 12 Oct 2005, 7:52 a.m.

I like the [Series 80 Emulator](#) because it let's me develop programs on my laptop (e.g. when I go to work by train). To transfer the files from the emulator to my HP-85 I worked out the following procedure:

- First you need a mass storage device in your Series 80 emulator. Copy one of the LIF disk images which come along with the emulator (e.g. FL86-DEMO.LIF) and rename it to XFER.LIF. Use command Edit/:D7xx/Load to make it accessible to the emulator and execute INITIALIZE "XFER",":D7xx" to delete all files.
- Because the Series 80 emulator is in fact an HP-87 and I have got an HP-85, I use [GETSAVE](#) to store the program as a string. The following statements assume GETSAVE is located on the volume UTIL.
LOADBIN "GETSAV.UTIL"
SAVE "MYPROG.XFER"
- Do not forget to save :D7xx to file XFER.LIF using Edit/:D7xx/Save. Now, you can quit the emulator and open a command shell.
- Extract MYPROG from the LIF file XFER.LIF. I use my own program to do this, but you may use the [LIF utilities for LINUX](#) (although I did not test that because I have no LINUX). It is also possible to do this manually with the help of a hex editor, but you have to know what you do.
- Start HP's LIF Utilities for PC (LIFUTIL.EXE) and select "DOS to LIF copy". Enter DOS directory and file name etc. In the "File format conversion" field, choose "Binary image: No format change, user-def. type". Press F6 to define LIF type and enter 0010 for type and 00000001 for implementation. Finally, copy the file to floppy disk by pressing F5.
- Insert the floppy disk in your HP-85 drive and load the program:
LOADBIN "GETSAV.UTIL"
GET "MYPROG.XFER"

This procedure works well on my system and should also work on other PC's provided that you can write SD floppy disks with LIFUTIL.

Juergen Keller

Many thanks to Olivier De Smet for developing the Series 80 emulator and to Vassilis Prevelakis and John Shadbolt for their great Series 80 web sites.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

SOLVE and INTEG on HP's RPN-based models

Posted by [Karl Schneider](#) on 2 Oct 2005, 12:05 a.m.

SCOPE:

This article summarizes the implementations of "SOLVE" and "INTEG" machine-coded functions on [RPN-based](#) Hewlett-Packard calculators. In addition, a procedure is presented for utilizing these functions on the HP-34C, HP-15C, and HP-41C/CV/CX with multiple-input, single-output (MISO) user-defined functions.

'SOLVE' AND 'INTEG' FUNCTIONS

AVAILABILITY:

On many of its RPN-based scientific calculators, Hewlett-Packard offered sophisticated numerical rootfinding ("SOLVE") and numerical integration ("INTEG") of single variables within general user-defined functions. These are built-in or made available on the following models:

Original implementation

- [HP-34C](#)
- [HP-15C](#)
- [HP-41C/CV/CX](#) with [Advantage Pac](#)

Later implementation

- [HP-32S](#)
- [HP-42S](#)
- [HP-32SII](#)
- [HP-33s](#)
- [HP-35s](#)

None of the following are addressed in this article, due to substantial differences in capabilities and protocol of usage:

- ROOT and INTEGRATE on the [RPL-based HP-28](#), [HP-48](#), [HP-49](#), and [HP-50g](#) series models
- The algebraic equation solver on the [HP-17B](#), [HP-18C](#), HP-19B, and [HP-27S](#) series models
- FNROOT and INTEGRAL on the [HP-71B](#) Math Pac
- Keystroke programs "SOLVE" and "INTG" on the HP-41 Math Pac
- Keystroke programs "ROOT" and "INT" on the algebraic-entry [HP-20S](#)
- PROOT polynomial-function solver on the HP-71B Math Pac and HP-48 series models

SELECTED REFERENCES:

The philosophy, methods, and effective utilization of 'SOLVE' and 'INTEG' were thoroughly described in the three user's manuals developed for the HP-34C and the HP-15C. These manuals are the only ones that provide exceptionally detailed insight into the 'SOLVE' and 'INTEG' functions. The two *Journal* articles were written for the HP-34C, but are also informative. Of course, the user should always refer to the pertinent Owner's Handbook for guidance in basic usage of 'SOLVE' and 'INTEG' on a particular model of calculator.

- Kahan, W. M., "Personal Calculator Has Key to Solve Any Equation $f(x) = 0$," *Hewlett-Packard Journal* 30:12, December 1979
- Kahan, W. M., "Handheld Calculator Evaluates Integrals," *Hewlett-Packard Journal*, 31:8, August 1980
- *HP-34C Owner's Handbook and Programming Guide*, Sections 8 and 9, Appendices A and B (spanning pp. 174-268)
- *HP-15C Owner's Handbook*, Sections 13 and 14, Appendices D and E (spanning pp. 180-204 and 220-258).
- *HP-15C Advanced Functions Handbook*, Sections 1 and 2 (spanning pp. 6-64)

For each of these references, a scan of a printed copy (in Adobe Acrobat "PDF" format) is present on [CD-ROM and DVD-ROM disks available from the MoHPC](#).

ORIGINAL IMPLEMENTATION:

The SOLVE and INTEG functions were introduced on the HP-34C in 1979, and later directly ported to the HP-15C in 1982. For the HP-41C/CV/CX, the same functions were adapted with only slight modification in the Advantage Pac introduced in 1985. Consequently, the protocol for using SOLVE and INTEG on these models is almost identical.

The user's mathematical function to be solved or integrated must be programmed as an RPN routine. The routine must be designed to accept a value of the "floating" input variable to be solved or integrated from the stack x-register, and to return the calculated function to the x-register. SOLVE or INTEG will fill the stack with a current "updated" value of the floating variable to be utilized by the user's program, which will then be automatically executed. This process will continue as necessary -- by SOLVE until a root within the calculator's limit of accuracy is found (or determined to be unobtainable), or by INTEG until the integral is calculated to the estimated accuracy that is justified by the user's specification.

Moreover, the program may include a SOLVE or INTEG instruction, thus allowing INTEG to be executed within SOLVE, or vice versa. This can be useful for solving a limit of integration, or a parameter of an integrand.

SOLVE and INTEG on the HP-34C, HP-15C, and HP-41C/CV/CX are ideally-suited for single input, single-output (SISO) user-defined functions. This is because these models provided only numbered and "indirect" storage registers, with no provision to assign alphabetical identifiers to variables.

LATER IMPLEMENTATION:

The mathematical algorithms of SOLVE and INTEG remained virtually unchanged in the Pioneer-series HP-32S and HP-42S introduced in 1988 and the HP-32SII introduced in 1991. However, the protocol for utilizing and invoking these functions was adapted for the new features and capabilities of the Pioneer-series models.

One new feature was alphabetical identifiers for variables. Storage registers for variables on the Pioneer-series HP-32S and HP-32SII (as well as its derivative HP-33s and HP-35s), are designated by single letters. The Pioneer-series HP-42S allowed full alphanumeric identifiers for variables.

On the HP-32S and HP-42S, the user's mathematical function to be solved or integrated must be programmed as an RPN routine. On the HP-32SII, HP-33s, and HP-35s, the user's function may be programmed as an RPN routine or entered as an algebraic expression. (Alternatively, on the HP-33s and HP-35s, the user's function may be programmed

as an algebraic-entry routine.)

The protocol of SOLVE and INTEG on each of these models is well-suited for multiple-input, single-output (MISO) user-defined functions. This is because SOLVE and INTEG pass values of the "floating" input variable to the user's function by its storage register, not by the stack x-register.

The user must specify (by letter or name) the root variable to SOLVE or the dummy variable of integration to INTEG. Upon invocation, SOLVE or INTEG will prompt the user for the fixed values of all other variables appearing in the algebraic expression or entered via an INPUT instruction. Finally, the user's RPN routine or algebraic expression must recall any input variable from its storage register.

Neither the HP-32S, the HP-32SII, the HP-33s, nor the HP-35s provide the capability to execute INTEG within SOLVE, or vice versa. The HP-42S, however, retained that capability of the original implementation.

On the HP-33s and HP-35s, a direct-solution method for algebraic expressions (seemingly adapted from the algebraic solver on the HP-17/18/19/27 models) was incorporated into SOLVE. When the form of the user's equation is such that one root can be determined by algebraic manipulation, the HP-33s and HP-35s will always return that solution, no matter what the initial first guesses are.

INTEGRATION: METHOD, DESIRED ACCURACY, AND ESTIMATE OF ERROR

'INTEG' uses a Romberg method for calculating the result of an integral. It takes samples that are not uniformly spaced, in order to suppress resonance or aliasing that can result from sampling periodic functions.

The developers of 'INTEG' recognized the impracticality of requiring the user to specify the number of evaluations of the integrand, as other implementations had done. Instead, 'INTEG' requires the user to specify an *uncertainty* or *accuracy factor* of the integrand. 'INTEG' will generally succeed at calculating the integral to an accuracy that is well within the estimate of maximum error derived from the given uncertainty and the range of integration.

On all of the RPN-based calculators, the calculated value of the integral is placed in the x-register; the estimated maximum error is placed in the y-register. This allows for easy summation of a subdivided integral and estimated error using the statistical Sigma+ function.

HP-34C, HP-15C, HP-41 Advantage Pac, HP-32S, HP-32SII:

On these calculators, the user specifies the uncertainty of the integrand by setting the display mode (i.e., FIX, SCI, ENG, or ALL). The premise is as follows: The uncertainty of the integrand is assumed to be half the resolution of the value displayed under the mode setting. In other words, the "true value" of the integrand is assumed to be *anywhere* within the range for which the displayed value is the correct rounded value.

FIX, SCI, and ENG are used with an argument up to 9 on the HP-34C, HP-15C, and HP-41; they are used with an argument up to 11 on all Saturn-processor models and their derivatives, including the HP-32S and HP-32SII.

The FIX display mode specifies an *absolute* uncertainty of 0.5×10^{-N} , where N is the lesser of the number of *displayed* or *displayable* digits following the decimal point. (Thus, an integrand value of -3141592.654 on the HP-34C, HP-15C, or HP-41 would carry a "FIX 3" uncertainty, even if specified as "FIX 4".) If all decimal digits of the integrand are displayable in "FIX N" format over the lower and upper limits of integration "a" and "b", the estimate of maximum error is simply $(b-a) * 0.5 \times 10^{-N}$.

The SCI and ENG display modes specify a *stepwise relative* uncertainty of $0.5 \times 10^{M-N}$, where N is the number of digits after the decimal point and M is the value of the base-10 exponent of the integrand, if displayed in the selected format. Thus, SCI and ENG give identical results for a given N. "ALL" -- on the HP-32S/SII models -- is equivalent to

"SCI 11" or "ENG 11".

When $M = 0$ (integrand value between 1 and 10), SCI and ENG specify the same uncertainty as FIX for a given N. When $M < 0$ (integrand value between 0 and 1), SCI and ENG specify a smaller uncertainty than FIX for a given N. The calculator must determine the estimate of maximum error by integrating the uncertainty formula over the set of sample points.

While it may not be considered ideal to utilize a display-mode setting to specify the numerical uncertainty of a function output, this "original" method is clever. Because *some* display-mode setting is always in effect, the user cannot omit the uncertainty parameter, which is necessary for specifying the scope of the calculation but never appears in the actual problem. Moreover, the display-mode settings most favored by users (e.g. FIX 5, SCI 3, ENG 4) will usually provide accurate answers without excessive evaluation of the integrand function.

HP-42S:

The HP-42S utilizes the approach introduced in the HP-71B Math ROM, in which the user must enter the uncertainty of the integrand as a relative per-unit "accuracy" parameter. This value, entered as the variable "ACC" in the HP-42S, is a multiplicative tolerance, whose operative value within 'INTEG' is restricted to positive real values up to and including unity ($0 < \text{"ACC"} \leq 1$). The "true" value of the integrand is taken to lie within a band whose width is $\pm \text{"ACC"}$ (as restricted) times the calculated value of the integrand.

Thus, the estimate of maximum error ("*uncertainty of computation*") of the calculated integral over the specified range will generally be within "ACC" times the integral of the absolute value of the integrand function over the same range.

This approach has definite advantages: The bandwidth of uncertainty is smooth -- directly proportional to the value of the integrand. Any value of "ACC" between zero and unity may be utilized, including those that correspond directly to the accuracy of an integrand that best fits measured data. This approach is practical on the HP-42S, because the required accuracy factor "ACC" is a named variable that need not be placed on the stack, and is difficult to overlook on the integration menu.

HP-33s and HP-35s:

The HP-33s and HP-35s also require the user to specify a proportional accuracy factor *à la* "ACC" in the HP-42S, but that value is defined by use of the display-mode settings FIX, SCI, ENG, and ALL. For example, FIX 3 specifies an accuracy factor of 0.001. The approach used in the HP-33s and HP-35s is the same one used for the RPL-based HP-48 series. Because display-mode settings must be used, the only permitted values of accuracy factor are of the form 10^{-N} , where N is the parameter provided to FIX, SCI, ENG, or ALL.

Thus, the estimated maximum error of integration is about 10^{-N} times the integral of the absolute value of the integrand. (ALL corresponds to $N = 11$.) FIX, SCI, or ENG will yield identical results for a given value of N.

TECHNIQUE: 'SOLVE' AND 'INTEG' WITH "MISO" USER FUNCTIONS ON THE 34C/15C/41C

In order to conveniently employ SOLVE or INTEG on the HP-34C/15C/41C for multiple-input, single-output (MISO) user-defined functions, the user's program should emulate the way SOLVE and INTEG manage input variables on the three Pioneer-series models and the HP-33s.

Assume that a MISO function is to be programmed as an RPN routine on a HP-34C, HP-15C, or HP-41C/CV/CX. A user may wish to use SOLVE or INTEG successively for any selected "focus" variable with all others held constant, and without having to edit the RPN routine. Here is a simple procedure for this purpose:

1. Choose a numbered storage register for each variable in the function.
2. On the 41C/CV/CX, choose another numbered register as the indirect storage register. (Do not use a stack register, as it will be overwritten by SOLVE or INTEG.)

In PROGRAM mode:

3. Program the function as an RPN routine that not only meets the basic requirements, but also retrieves each variable from its chosen storage register when it is used. (On the 41C/CV/CX, the label beginning the routine must be an external label.)
4. The second instruction of the routine (immediately after LBL) should store the stack x-register value to the indirectly-referenced register. This instruction is "STO (i)" on the HP-34C and HP-15C; it is "STO IND nn" on the HP-41C/CV/CX.

In RUN mode:

5. Store the desired constant values of the fixed-value variables to their storage registers.
6. Store the *register number* containing the floating variable to the indirect storage register.
7. Invoke SOLVE or INTEG in the usual manner.

This procedure utilizes indirect storage to make the RPN program more flexible. SOLVE and INTEG feed the each value of the floating variable as input to the user program, which immediately stores the value indirectly to its user-chosen register. Each variable in the function is then recalled for use in calculations, so the user program need not be structured to receive any particular variable from the stack.

EXAMPLE:

$$f(x, y, z) = 2 \cdot x - \ln y + 1/z$$

x in R1; y in R2; z in R3
indirect in R00 (HP-41 only)

HP-15C/HP-34C program:

```
LBL A
STO (i)
RCL 1
2
*
RCL 2
LN
-
RCL 3
1/x
+
RTN
```

HP-41C/CV/CX program:

```
LBL "AA"
STO IND 00
RCL 01
2
*
RCL 02
LN
-
RCL 03
1/x
+
RTN
```

In RUN mode, solve for x such that $f(x, y=15.1, z=5.3) = 0$

HP-15C/HP-34C:

```
15.1
STO 2
5.3
STO 3
1
STO I
0.5
```

HP-41C/CV/CX:

```
15.1
STO 02
5.3
STO 03
1
STO 00
0.5
```

ENTER
5
SOLVE A

ENTER
5
"AA" (enter AA in alpha mode)
XEQ "SOLVE"

(Answer is 1.263007749)

Next, solve for y such that $f(x=0.7, y, z=3.3) = 0$

HP-15C/HP-34C:

0.7
STO 1
3.3
STO 3
2
STO I
0.5
ENTER
6
SOLVE A

HP-41C/CV/CX:

0.7
STO 01
3.3
STO 03
2
STO 00
0.5
ENTER
6
"AA" (enter AA in alpha mode)
XEQ "SOLVE"

(Answer is 5.490560270)

Finally, integrate $f(x=1.7, y=4.1, z)$ over $z = 1.0$ to 6.0 .
Specify 5 decimal-digit absolute function accuracy:

HP-15C/HP-34C:

FIX 5
1.7
STO 1
4.1
STO 2
3
STO I
1
ENTER
6
INTEG A

HP-41C/CV/CX:

FIX 5
1.7
STO 01
4.1
STO 02
3
STO 00
1
ENTER
6
"AA" (enter AA in alpha mode)
XEQ "INTEG"

(answer is 11.73682)

Edited: 12 Dec 2010, 5:54 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-IL, Linux and the Internet

Posted by [Howard Owen](#) on 27 July 2005, 4:07 p.m.

Introduction

In terms of computer networks, HP-IL is almost as retro as you can get. It's not as old as [the ARPANET](#), but it's older than that network's transition from [NCP](#) to [TCP-IP](#), an event that many consider to be the birthday of the Internet. HP-IL is also contemporaneous with [DECNET](#) phase II, [ARCNET](#) and [Ethernet](#) (depending on who you talk to).

It is, in any event, a very old networking stack. It's so old, that it's design probably predates the invention of the term "networking stack." (Though I can't find a reference to the first use of that phrase, HP-IL certainly predates the OSI layer model. It doesn't predate the idea of layering protocols in computer networks, however.) It is old enough that getting it working with a modern protocol suite is a challenge.

Topic Sentence (a little late)

This is a quick implementation note describing how I set up Christoph Klug's HP-IL ISA card, JF Garnier's EMU41 and Slackware Linux to talk to each other. The network I built is depicted in the following diagram:



Hardware

The PC is a Compaq Deskpro 575. This machine isn't quite as old as some of the other components, but it qualifies as "retro" also. It features

- Pentium CPU @ 75MHz. (That's 0.075 GHz for the youngsters)
- Lots of ISA slots (5)
- A couple of PCI slots (2)
- Onboard 100/baseT ethernet networking.
- Adequate, though cramped disk space. (600MB)
- Reasonable RAM (196MB, of which only 128MB are working at the moment. Hmm.)

The critical stat is the number of ISA slots. In addition to Christoph's HP-IL board, I have two ISA HP-IB cards and an original HP 82793 HP-IL card. I can theoretically have them all plugged in at once on this system.

Software

The OS running on this system is Slackware Linux 10.1. I chose this distribution because it was the one current Linux distro that *nearly* made it on to the Compaq when there was only 16MB of RAM installed. It also had the smallest installed size for a modern distro without X11.

[FreeDOS](#) is installed on a 80MB partition at the start of the disk. This MS-DOS clone is perpetually in beta. The version I installed is the latest release, 0.9 beta. I haven't tried to do anything fancy in the DOS environment because most of that is supplied in Linux. (See below)

[EMU41](#) is installed in the FreeDOS partition. This is the registered version that supports Cristoph Klug's ISA card, as well as the HP82973A card. This software makes a lot of what I do in this environment possible.

[DOSEMU](#) is a virtual machine for DOS that runs under Linux. A VM is required since DOS programs like to bang on the hardware directly. Unix-like operating systems such as Linux don't approve of that sort of thing, so DOSEMU provides a software layer that DOS can abuse, while protecting the real hardware from harm. I have version 1.22 of DOSEmu installed under Slackware.

Edited: 27 July 2005, 7:50 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

A Unix Geek Rediscovered the HP-41

Posted by [Howard Owen](#) on 5 June 2005, 8:08 p.m.

The Good Ol' Days

Back in 1982, I was attending Santa Barbara City College studying first year chemistry. I discovered the HP-41C in the campus bookstore and fell in love with it. I scraped some tutoring money together to buy the basic model and started playing with the machine. I learned RPN and programming at the same time. I'm sure the experience warped my impressionable student's brain, because for several years afterwards, I couldn't really grasp a difficult algorithm or programming challenge without recoding it, at least in my head, into the HP-41C's register oriented, assembler-like programming language. As further evidence of the HP-41C's insidious influence, I switched majors to Computer Science within a year of obtaining the calculator. I attribute this partly to the fun and success I had programming the 41-C.

Way Too Much Time Passes

23 years later, in 2005, I had become a successful Unix/Linux geek with 20 years of experience. I was still in love with gadgets, as any self respecting geek must be, but the old 41-C was long gone, having been stolen from me while I napped under the shade of a campus tree back in the bucolic 1980s. My current gadget fascination was with the Zaurus line of Linux based handheld computers from Sharp. I was working on one of the many Linux distributions that had sprung up for those machines. This particular "distro", called [pdaXrom](#), was based on the X11 windowing system used by larger Linux machines. Part of the attraction this held for me is that software that runs on the larger machines should be more easily ported to pdaXrom than to some of the other Zaurus systems that weren't so based. I was looking for something to port, and I noticed that the default calculator in pdaXrom was lame. (All calculators are lame, in one way or another, after you have gotten used to a top-of-line HP.) I started casting around for a better calculator with available source code, and quickly came across Eric Smith's [Nonpareil](#). This software emulated many old HP machines, including the HP-41CV. (Eric now has the CX code working as well.) The software looked great while running, and it was based on recent Linux X11 toolkits, the very ones that pdaXrom implemented, as a matter of fact. This looked promising, and so it proved. It took me about two hours to whip up a script to build and package just the HP-41CV calculator for my first release to pdaXrom.

Fatal Attraction

Along the way, I felt that old, fatal attraction returning. Playing with Nonpareil's HP-41CV simulation was great fun, but frustrating as well. It lacked the tactile feedback of that great keyboard, for one thing. It was also slow tapping in keystrokes with a stylus or a mouse. I needed the **real thing!** Fortunately, there was this newfangled thing called eBay. Soon I had a real HP-41CX actually resting in my hands! I was delighted. I immediately set to work coding a card deck program on my old/new toy.

The Morning After

I quickly came to realize something that wasn't part of my rosy memory of the machine. Application programming on an HP-41 is *hard*. It really is more like assembler than any higher level language. Only, there's no macro facility or aliases for cryptic addresses like you get with a decent macro assembler. the 41-C's registers are numbered from zero to SIZE-1. In a large program, how do I remember which register I put the number of decks in? Was it in 12 or 22? Scrolling back through a long listing on the 41-C is painfully slow, at least for someone who has survived the last twenty years of growth in computer power, so searching for assignments that way was hard. Without a printer to view the whole listing, any program of non-trivial size quickly becomes difficult to manage on the calculator itself.

Searching for Solutions

The problem of connecting an HP-41C to a PC is an interesting one. But I'm still pursuing answers as I write this. More immediately, I came up with some ideas to get around the coding difficulty issue, assuming I was able to get text or compiled code exchanged between the calculator and my desktop. One idea is to define symbols for registers and labels using the standard syntax of the C preprocessor (cpp.) The symbols can then be stripped out by the C preprocessor before the text code is compiled for download to a real HP-41C or to an emulator. I like having the line numbers in my listings, because it makes it easier to work back and forth between the calculator and the listing. But when you are working in a text editor, there's no automatic renumbering when you delete and add lines. So I wrote a Perl script to do the renumbering. Finally, I wrote another Perl script to remove comments and line numbers from the output of the C preprocessor. I also added another one to add DOSish line endings to the output. And thereon hangs a tale.

Emulation: HP-41 on DOS on Linux

I use [Jean-François Garnier's excellent EMU41 emulator](#) in order to prepare the way for the day I plug in [Cristoph Klug's](#) clone of the HP 82973A HP-IL to PC interface. But JF's software is designed to run in MS-DOS. Not a problem! It works just fine using the great [DOSEMU virtual machine](#) for Linux, which includes the [FreeDOS](#) MS-DOS replacement software. As a bonus, the [41UC tools](#) also work well in the emulated DOS environment.

An Example

The following program isn't intended to be the last word in card decks for the HP-41C. I wrote it to test out my preprocessing process before uploading it to EMU41. The first listing is the source form, complete with cpp definitions, comments and symbolic names for registers, labels and constants. The second is a portion of the file prepared for processing by 41UC.EXE into a form that EMU41 can load.

```
# Storage registers
#define nSuits 00
#define nCards 01
#define Suit 02
```

```
# Labels
#define rankLoop 00
#define suitLoop 01
#define Ace 02
#define tenJQK 03
#define Ten 04
#define Jack 05
#define Queen 06
#define Clubs 07
#define Hearts 08
#define Diamonds 09
#define stoSuit 10
#define addCard 11
```

```
# Character constants
```

```

#define charZero 48
#define charOne 49
#define charA 65
#define charC 67
#define charD 68
#define charH 72
#define charJ 74
#define charK 75
#define charS 83

# Other variables
#define deckRegisterOffset 15

01 LBL "INIDECK"
05 4
06 STO nSuits # number of suits
07 LBL rankLoop
08 13
09 STO nCards
10 LBL suitLoop
11 RCL nSuits # alpha code the suit
12 1
13 X=Y?
14 GTO Clubs # Clubs
15 RDN
16 2
17 X=Y?
18 GTO Hearts
19 RDN
20 3
21 X=Y?
22 GTO Diamonds
23 charS # ASCII S, Spades, by default
24 GTO stoSuit
25 LBL Clubs
26 charC # ASCII C
27 GTO stoSuit
28 LBL Hearts
29 charH # ASCII H
30 GTO stoSuit
31 LBL Diamonds
32 charD # ASCII D
33 LBL stoSuit
34 STO Suit # ASCII suit
35 RCL nCards # Encode the rank
36 1
37 X=Y? # Card is an ace
38 GTO Ace
39 RDN
40 9
41 X<Y? # card is 10, J , Q or K
42 GTO tenJQK
43 RDN # Card is 2-9
44 charZero # we can add 48
45 + # to get ASCII
46 GTO addCard

47 LBL Ace # Card is an ace
48 65 # ASCII A
49 GTO addCard

50 LBL tenJQK # 10, J, Q or K
# Figure out which
51 RDN
52 10
53 X=Y?
54 GTO Ten
55 RDN
56 11 # Jack
57 X=Y?
58 GTO Jack
59 RDN
60 12 # Queen
61 X=Y?
62 GTO Queen
63 charK # Must be a King

```

```

64      GTO addCard

65 LBL Ten #
66      charOne # two characters
67      XTOA
68      charZero
69      GTO addCard

70 LBL Jack
71      charJ
72      GTO addCard
73 LBL Queen # Q
74      charQ

75 LBL addCard # Add the card to Alpha
76      XTOA # First add the rank in X
77      RCL Suit # Then the suit
78      XTOA

# Compute the storage register this card will go in.

79      RCL nSuits # Suits left
80      INT # It's a loop counter
80      1 # Subtract 1 for
81      - # the current suit
82      13 # this many cards in each remaining suit
83      *
84      RCL nCards # Cards left in this suit
85      INT # It's a loop counter also
85      +
86      deckRegisterOffset # Add this to the calculated address
87      +
88      VIEW X # Keep track of progress
89      ASTO IND X
90      CLA
91 DSE nSuits
92 GTO suitLoop
93 DSE nCards
94 GTO rankLoop

```

Here is the result of running the following pipeline on Linux with the above file as input. This example uses the bash shell.

```

% perl 41pp.pl < deck.txt | cpp -P | perl crlf.pl | head -10
LBL "INIDECK"
4
STO 00
LBL 00
13
STO 01
LBL 01
RCL 00
1
X=Y?

```

The first command in the pipeline is the perl script whose source is given below. It strips out all comments except for the #define lines that the c preprocessor will use. The script also removes line numbers and leading spaces from the pipeline

The next command in the pipeline is the C preprocessor. We give it the -P option, which causes it to not emit certain control information that it normally does for the gcc compiler. The cpp processes the #defines replacing the first given text with the second. This strips the symbols from the source, replacing them with the correct numeric equivalents.

Next we run the data through a filter that ensures that the line endings are in DOS format. Then we finish up by piping the output through the head command, to give us a look at the first ten lines. Normally we would redirect the output to a file, which would then be ready to feed to HP41UC.EXE for conversion to whatever format we wanted.

Here are the two Perl filters from the above pipeline:

```
#!/usr/bin/perl
# 4lpp.pl
# This script is Free Software. You may redistribute it under the terms
# of the GNU GPLv2 as found at http://www.fsf.org/licensing/licenses/gpl.txt
```

```
while(<>){
    chomp; # remove the newline
    s/\r$//; # and the CR too.
    print "$_\n" if (/^\#\s*define/); # special case these comments
    next if (/^\s*$ / or /^#/); # Skip blank or comment lines
    s/^\d+//; # Strip line numbers
    s/^\s+//; # Strip any leading space
    s/\s*#.*/; # Strip trailing comments
    print "$_\n";
}
```

```
#!/usr/bin/perl
# crlf.pl
# This script is Free Software. You may redistribute it under the terms
# of the GNU GPLv2 as found at http://www.fsf.org/licensing/licenses/gpl.txt
```

```
while(<>){
    chomp; # remove any newline
    s/\r$//; # and any CR too.
    print "$_\r\n"; # Put them back the way we want
}
```

[Edit Message](#)
[Delete Message](#)

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

Fast and Accurate Trigonometric Functions on the HP-12C

Posted by [Gerson Washiski Barbosa](#) on 22 Apr 2005, 2:33 a.m.

Keystrokes	Display	Comments
[f] [P/R]		
[f] CLEAR [PRGM]	00 -	
3	01 - 3	SIN_COS
[/]	02 - 10	
[STO] [n]	03 - 44 11	
ENTER	04 - 36	
[x]	05 - 20	
ENTER	06 - 36	
ENTER	07 - 36	
ENTER	08 - 36	
RCL] 6	09 - 45 6	
[x]	10 - 20	
RCL] 5	11 - 45 5	
[+]	12 - 40	
[x]	13 - 20	
RCL] 4	14 - 45 4	
[+]	15 - 40	
[x]	16 - 20	
RCL] 0	17 - 45 0	
[+]	18 - 40	
RCL] [n]	19 - 45 11	
[x]	20 - 20	
ENTER	21 - 36	
ENTER	22 - 36	
[x]	23 - 20	
4	24 - 4	
[x]	25 - 20	
[CHS]	26 - 16	
3	27 - 3	
[+]	28 - 40	
[x]	29 - 20	
ENTER	30 - 36	
ENTER	31 - 36	
ENTER	32 - 36	
[x]	33 - 20	
[CHS]	34 - 16	
1	35 - 1	
[+]	36 - 40	
[g] [SQRT]	37 - 43 21	
[g] [GT0] 00	38 - 43, 33 00	ASIN
ENTER	39 - 36	
[x]	40 - 20	
[1/x]	41 - 22	
1	42 - 1	
[-]	43 - 30	
[1/x]	44 - 22	
[g] [GT0] 51	45 - 43, 33 51	ACOS
ENTER	46 - 36	
[x]	47 - 20	
[1/x]	48 - 22	
1	49 - 1	
[-]	50 - 30	
[g] [SQRT]	51 - 43 21	
[g] [x=0]	52 - 43 35	ATAN

[g][GT0]00	53-	43,33	00	
1	54-		1	
[x<>y]	55-		34	
[g][x<=y]	56-	43	34	
[g][GT0]60	57-	43,33	60	
[1/x]	58-		22	
9	59-		9	
0	60-		0	
[ST0][FV]	61-	44	15	
[RDN]	62-		33	roll down key (Rv)
[1/x]	63-		22	
[ST0][n]	64-	44	11	
[ENTER]	65-		36	
[x]	66-		20	
+	67-		40	
[g][SQRT]	68-	43	21	
[RCL][n]	69-	45	11	
-	70-		30	
[ST0][n]	71-	44	11	
[ENTER]	72-		36	
[x]	73-		20	
[ENTER]	74-		36	
[ENTER]	75-		36	
[g][12/]	76-	43	12	
[ENTER]	77-		36	
[RCL]3	78-	45	3	
+	79-		40	
[x]	80-		20	
[RCL]2	81-	45	2	
+	82-		40	
[x]	83-		20	
[RCL]1	84-	45	1	
+	85-		40	
[x]	86-		20	
1	87-		1	
+	88-		40	
[RCL][n]	89-	45	11	
[x]	90-		20	
2	91-		2	
[x]	92-		20	
[RCL]0	93-	45	0	
/	94-		10	
[RCL][FV]	95-	45	15	
[g][x=0]	96-	43	35	
x<>y	97-		34	
x<>y	98-		34	
-	99-		30	
[f][P/R]				

(Listing generated by Tony Hutchin's 12c.exe utility)

After entering the program, store the following constants:

R0: 1.745329252E-02 (pi/180, for angle conversions)

R1: -3.333324820E-01
R2: 1.998722722E-01 (a, b and c coefficients for the
R3: -1.397054038E-01 arctangent polynomial)

R4: -8.860960894E-07
R5: 1.349561365E-11 (a, b and c coefficients for the
R6: -9.716369450E-17 sine polynomial)

Here is a state file for [Nonpareil](#) so the program can be tried out on the simulator before one decides to key it in:

[12ctrigs](#)

As the state file was saved with no extension, some browsers may add a .txt extension to it. The file extension may be removed or left unchanged (the current version of Nonpareil will accept it just the same).

Usage:

Trigonometric functions:

Enter angles in degrees, $0 \leq x \leq 90$:

```
cos(x): [R/S]
sin(x): [R/S] [x<>y]
tan(x): [R/S] [/]
```

Maximum absolute error: about $5E-10$ ($\sin(x)$).

Inverse trigonometric functions:

Enter arguments within the following ranges:

```
0 < x < 1          for arcsin(x);
0 < x <= 1         for arccos(x); (*)
1E-50 < x < 1E50   for arctan(x)  (**)
```

Results are given in degrees

(*) $\cos(x)$ and $\tan(x)$ functions are not accurate near the 90 degree boundary because $\cos(x)$ is computed as a function of $\sin(x)$ at that same region, in which the 10 digit accuracy of the calculator is not enough to make distinction between different arguments close to that limit. Results given by the sine function, on the other hand match to 9 decimal places those given by the HP-15C, over the range $5.729577E-47 \leq x < 90$. As a workaround, $\cos(x)$ can be calculated as $\sin(90-x)$, if $\cos(x)$ or $\tan(x)$ is needed to full accuracy.

(**) These are the valid ranges, not the ranges where accuracy is granted. Arguments below $1E-05$ or above $1E04$ for the $\arctan(x)$ are subject to rounding errors.

```
arcsin(x): [g][GT0]39 [R/S]
arccos(x): [g][GT0]46 [R/S]
arctan(x): [g][GT0]52 [R/S]
```

The constant stored in $R0$ ($\pi/180$) may be used to convert angles from degrees to radians and vice-versa.

The key pattern below makes remembering the entry-points addresses much easier:

[4] [5] [6]

[2] [3]

vertical column at right, keys [3] and [9]: $\arcsin(x)$ (think of the sine-axis)
middle horizontal row, keys [4] and [6]: $\arccos(x)$ (think of the cosine-axis)
central remaining keys [5] and [2]: $\arctan(x)$

Maximum relative error: about 4.2E-06 %

Some results: (compared with exact results at this precision)

	<i>sin</i>	<i>cos</i>	<i>tan</i>
0.0001	0.000001745 (0.000001745)	1.000000000 (1.000000000)	0.000001745 (0.000001745)
1.0000	0.017452406 (0.017452406)	0.999847695 (0.999847695)	0.017455065 (0.017455065)
15.0000	0.258819045 (0.258819045)	0.965925826 (0.965925826)	0.267949192 (0.267949192)
30.0000	0.500000000 (0.500000000)	0.866025404 (0.866025404)	0.577350270 (0.577350269)
45.0000	0.707106781 (0.707106781)	0.707106781 (0.707106781)	1.000000000 (1.000000000)
60.0000	0.866025403 (0.866025404)	0.500000001 (0.500000000)	1.732050804 (1.732050808)
89.0000	0.999847695 (0.999847695)	0.017452415 (0.017452406)	57.28993250 (57.28996163)
89.9900	0.999999985 (0.999999985)	0.000175499 (0.000174533)	5698.028737 (5729.577893)
89.9999	1.000000000 (1.000000000)	0.000020000 (0.000001745)	49,999.9999 (572,957.7951)
	<i>asin</i>	<i>acos</i>	<i>atan</i>
0.0001	0.005729578 (0.005729578)	89.99427042 (89.99427042)	0.005729578 (0.005729578)
0.1000	5.739170147 (5.739170477)	84.26082985 (84.26082952)	5.710593018 (5.710593137)
0.5000	29.99999978 (30.00000000)	60.00000022 (60.00000000)	26.56505044 (26.56505118)
0.9999	89.18970794 (89.18970856)	0.810292057 (0.810291437)	44.99713700 (44.99713507)
1.0000	Error 0 (90.00000000)	0.000000000 (0.000000000)	45.00000190 (45.00000000)
2.0000	-	-	63.43494955 (63.43494882)
50,000	-	-	89.99885408

Running times:

This program does not intend to be either useful or practical, as it depends on constants being stored in all the remaining registers (If the sigma+ key is pressed accidentally, all constants will have to be entered again, just to mention a minor possible trouble). It also uses two financial registers as temporary variables. This should be regarded as a study for later implementation of these functions on the HP-12C Platinum, for example, whose greater memory capacity will allow for some improvements, such as exceptions handling, increase of the accuracy of the arctangent function and built-in constants.

(i) $\sin(3x) = \sin(x) * (3 - 4 \sin^2(x))$

(ii) $\sin(x) = x(1 + x^2(a + x^2(b + c \cdot x^2)))$, (x in radians)

with coefficients a , b and c optimized for the range $[0..\pi/6]$

(as angles are supposed to be entered in degrees, the constants 1, a, b and c are multiplied by successive odd powers of $\pi/180$),

(iii) $\arctan(x) = 2 \cdot \arctan\left(\frac{\sqrt{1/x^2 + 1} - 1/x}{2}\right)$, $0 < x \leq 1$ and

$$\arctan(x) = \frac{\pi}{2} - 2 \arctan(\sqrt{x^2 + 1} - x), \quad x > 1$$

(these are divided by the conversion factor $\pi/180$ so that results are given in degrees)

(iv) $\arctan(x) = x(1 + x^2(a + x^2(b + x^2(c + dx^2))))$

with a fixed coefficient ($d=1/12$) and a , b and c optimized for the range $[0..\sqrt{2}-1]$

(v) $\sin(x) = \sqrt{1 - \cos(x)^2}$)

(vi) $\tan(x) = \sin(x)/\cos(x)$

(vii) $\arcsin(x) = \arctan(\sqrt{1/(1/(x^2)-1)})$

(viii) $\arccos(x) = \arctan(\sqrt{1/(x^2)-1})$

The expression (iii) was derived from:

(ix) $\arctan(2a/(1-a^2))=2*\arctan(a)$

The expressions (i) and (iii) have been used here in order to reduce the usual $[0..pi/2]$ sine range and $[0..1]$ arctangent range to $[0..pi/6]$ and $[0..sqrt(2)-1]$, respectively, so that accuracy over those ranges is achieved with less constants.

Just for an idea of the importance of these range reductions, a 7th order minimax polynomial to approximate $\sin(x)$ over the range $[0..pi/4]$ yields a maximum absolute error of $2.6e-9$ (see reference 1 below). The 7th order polynomial approximation used in the program yields a maximum absolute error of about $9.5e-11$. Rather than mathematically determining the minimax polynomial coefficients, they were obtained numerically, by means of curve fitting (polynomial model, 100 to 150 samples), enough for practical purposes.

In case the registers are accidentally cleared, reentering only the angle conversion factor ($3.141592654/180$) in register 0 will allow for very rough approximations that might be enough, however, for certain applications.

References:

1) For some methods to approximate trigonometric functions, refer to the following paper, by Robin Green of Sony Computer Entertainment America:

Faster Math Functions

http://www.research.scea.com/research/pdfs/RGREENfastermath_GDC02.pdf

2) Expression (ix) was found at:

<http://functions.wolfram.com/01.14.16.0067.01>

Edited: 3 Aug 2005, 9:47 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

I have a flashing star (or snowflake) on the bottom left of my 1xc's display. What might it be?



The Museum of HP Calculators

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

I have a flashing star (or snowflake) on the bottom left of my 1xc's display. What might it be?

Posted by [Etienne Victoria](#) on 14 Apr 2005, 1:19 p.m.

Peace on you Lucky Brother !

When thou shalt see the Flashing Star on thy Display,

Thou shalt change those Holy Batteries in thy Calc.

And thou shalt not bother anymore,

For this Phenomenon shall appear,

No more often than the Halley Comet.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

Frequency

Posted by [Eric Smith](#) on 11 Apr 2005, 4:32 a.m.

As I was searching for something else entirely amongst old issues of 65 Notes and PPC Journal, I came across this gem. In addition to serving as an example of a useful programming technique for storing many binary values (flags) in a single numeric register, I thought it might also prove to be of practical value to some MoHPC readers.

Frequency

[reprinted from 65 Notes, October 1977, V4N8P7]

As my wife and I sat in our living room one fall evening, the kids all in bed, the conversation somehow evolved around to a discussion of the frequency of the marital relationship in our household. As the conversation progressed, it became quite apparent to me, that my wife was under the false impression that it was much more frequent than I knew it to be. I finally suggested, as a solution to this disagreement, that we "buy a calendar for our bedroom wall and circle all days on which activity occurred." My wife, being a little perturbed by now, stated very sarcastically, "Isn't there a way you can put that in your calculator?" That statement only shows some of her animosity toward her main competitor for my time and affection, namely my HP-67 pocket calculator.

After thinking about what she'd said for a day or so, it occurred to me that it really wasn't a bad idea. After all what a great opportunity to demonstrate to her how useful the HP-67 was in solving common household problems.

Researching my back issues of "65 Notes", I found Jake Jacobs (99) article on "Storing 33 Flags in One Register" - (V3N9P11). Oh! Just what I needed! With a little work, the following program resulted. I call it "Frequency".

Enter a date (mm.dd) on which activity occurred and push "A" to enter it into the calculator memory. If you make a mistake entering, enter the same data (mm.dd) and hit "B" to erase it. To see the activity for any given date, enter the date (mm.dd), push "C" and receive a "yes" or "no" answer. To get the complete listing of the activity for a given month, just enter the month number and hit "D". Enter a date (mm.dd) and hit "E" to get the last prior date on which activity occurred. (It can take approximately three minutes to search twelve months if no activity is recorded). The program is carefully written to take up only one side of one card, so program can be on one side and the record of one years "activity" can be stored on the other side of the same card.

My wife was, needless to say, thrilled to see such a unique solution to such a difficult problem. Although she's never actually said it, I do get the feeling that on one of her more trying days, she'd like to not only say "no" but add "and you can stick that in your calculator."

I sent a copy of this story to Lou Cargile, Jr. (753) as I thought he might get a kick out of it. He answered with, and I quote, "Could the program be expanded to make it possible to rate each entry on a scale of 1 to 10 so as to keep a log of quality as well as quantity? I guess this would quickly require two 67's at each bedside---disagreements will arise---but think of the potential market."

An excellent suggestion I thought, but since I haven't had the time to explore that possiblity, I leave that challenge up to the membership.

A footnote to this story: I hate to admit that I didn't discover this glitch in the program through actual use, but as I was checking out the program for publication, I found that when you try to record activity for all thirty one days of the month, the entry of the thirty-first day bombs the record of activity stored for the first, eleventh, and twenty-first. Realizing the devastating effect this constraint places on the "marital relationship", once again I contact that HP Wizzard of Wizzards, Lou Cargill, Jr. and here is his authoritative evaluation of the situation, and again I quote---"I think the calculator's handling of 31 days of consecutive "activity" in one month is probably jealousy, pure and simple. Do you suppose the calculator, like a mistress, is willing to share you, but just run up 31 days without one day of undivided attention and this occurs?"

Enough said---elsewhere in this issue you will find my program titled, "FREQUENCY".

Author's name withheld at wife's request(?) (1507)

Step	Key Entry	Key Code
1	LBL A	31 25 11
	GSB C	31 22 13
	x/=0?	31 61
	RTN	35 22
5	2	02
	RCL 0	34 00
	1	01
	-	51
10	y^x	35 63
	STO + i	33 61 24
	RTN	35 22
	LBL B	31 25 12
15	GSB C	31 22 13
	x=0?	31 51
	RTN	35 22
	2	02
20	RCL 0	34 00
	1	01
	-	51
	y^x	35 63
25	STO - i	33 51 24
	RTN	35 22
	LBL C	31 25 13
	GSB 4	31 22 04
30	LBL 3	31 25 03
	STO 0	33 00
	RCL i	34 24
	2	02
35	RCL 0	34 00
	3	03
	2	02
	x<=y?	32 71
40	COS	32 63
	Rdn	35 53
	y^x	35 63
	/	81
45	FRAC	32 83
	.	83
	5	05
	x>y?	32 81
49	CLx	44
	RTN	35 22
	LBL D	31 25 14
	GSB 4	31 22 04
50	1	01
	LBL 9	31 25 09
	GSB 3	31 22 03
	RCL 0	34 00
55	+	61
	PAUSE	35 72
	1	01
	STO + 0	33 61 00
	RCL 0	34 00
	GTO 9	22 09
	LBL 4	31 25 04

	1	01
	0	00
	x<>y	35 52
	x>y?	32 81
60	+	61
	DSP 1	23 01
	STO I	35 33
	FRAC	32 83
	EEX	43
65	2	02
	*	71
	RTN	35 22
	LBL E	31 25 15
	GSB 4	31 22 04
70	1	01
	-	51
	GTO 6	22 06
	LBL 5	31 25 05
	x<>I	35 24
75	LBL 6	31 25 06
	GSB 3	31 22 03
	x/=0?	31 61
	GTO 2	22 02
	RCL 0	34 00
80	x<>I	35 24
	DSZ	31 33
	GTO 5	22 05
	x<>I	35 24
85	LBL 7	31 25 07
	DSZ	31 33
	GTO 8	22 08
	2	02
	2	02
90	STO I	35 33
	LBL 8	31 25 08
	RCL i	34 24
	x=0?	31 51
	GTO 7	22 07
95	3	03
	1	01
	GTO 6	22 06
	LBL 2	31 25 02
	1	01
	0	00
100	RCL I	35 34
101	INT	31 83
	x>y?	32 81
	-	51
105	ABS	35 64
	RCL 0	34 00
	EEX	43
	2	02
	/	81
	+	61
110	DSP 2	23 02
111	RTN	35 22

Step	Instructions	Input Data/Units	Keys	Output Data/Units
1	Load side one			
1a	Load side two if you have previously recorded some data on it.			
2	Enter date (mm.dd) on which activity occurred.			
Please note:				
day must be				
3	entered as .03 or .04 not just .3 or .4	mm.dd	"A"	ignore
	If you make a mistake or to remove any date from memory, enter the date (mm.dd)	mm.dd	"B"	ignore
4	To check on the activity for any given date. Enter the date (mm.dd)			
	A ".5" means yes, A "0.0" means no entry was made for that date.	mm.dd	"C"	0.0 - "no" 0.5 - "yes"
5	Enter a month number (Jan is 1, Feb is 2, etc.). "D" will give a listing for entire month. Output "a" is day number			

6

"b" is - yes if .5; no if .0
Enter date(mm.dd) and "E" will locate
next prior date. Output - "a.b"
where "a" is month number and "b" is day
Please note:
This program is written to take only
one side of a card so the other side
can be used to store one year's data.
After you've entered your information
hit "w/data" and store it. Also you
must use "w/data" and pass thru
side 2 after each updateing of info.
If you try to enter a 'date' larger than
".31" you will get "Error". (example
3.5 will cause it to "Error" as you
are really saying 3.50.

mm.

mm.dd

"D"

"E"

"a.b"

"a.b"

Registers:

0 Day #	1 Jan.	2 Feb.	3 Mar.	4 Apr.
5 May.	6 Jun.	7 Jul.	8 Aug.	9 Sept.
A Oct.	B Nov.	C Dec.	D	E

I month

Labels:

A store	B erase	C test	D list	E prior
0	1	2 display date	3 check	4 initialize
5 decrementing months and days	----			9 month

Edited: 11 Aug 2011, 1:03 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

A Narrative History of HP from 1958-1990

Posted by [John Minck](#) on 5 Apr 2005, 4:07 p.m.

I posted this information on the other forum on this site too.

I recently expanded my Narrative History of HP to about 80 pages. It looks at my 37 years with HP during the "golden years" of high tech. It covers personalities, products and organization process, with looks at some of the legendary characters and products of HP.

It may be more than you wanted to know about H and P and me, but here is the URL.

[View the Narrative](#) (PDF).

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]
[Previous | Next]

Hyperbolic Approximations for Trigonometric Functions on the HP-12C

Posted by [Gerson Washiski Barbosa](#) on 26 Feb 2005, 1:27 p.m.

COS/SIN:	01 ENTER	ACOS:	52 ENTER
	02 ENTER		53 *
	03 ENTER		54 1/x
	04 68.3393		55 1
	11 /		56 -
	12 g e^x		57 g SQRT
	13 ENTER	ATAN:	58 1
	14 1/x		59 x<>y
	15 +		60 g x=<y
	16 CHS		61 g GT0 65
	17 4		62 1/x
	18 +		63 CHS
	19 x<>y		64 9
	20 ENTER		65 0
	21 *		66 x<>y
	22 21813		67 g e^x
	27 /		68 ENTER
	28 g e^x		69 1/x
	29 2		70 -
	30 *		71 g LSTx
	31 /		72 ENTER
	32 ROLLDN		73 1/x
	33 *		74 +
	34 g SQRT		75 g LSTx
	35 /		76 g LN
	36 x<>y		77 ROLLDN
	37 ENTER		78 /
	38 *		79 1844.6
	39 CHS		85 *
	40 1		86 ROLLDN
	41 +		87 ROLLDN
	42 g SQRT		88 ROLLDN
	43 *		89 6
	44 x<>y		90 y^x
ASIN:	45 g GT0 00		91 g SQRT
	46 ENTER		92 CHS
	47 *		93 32.2
	48 CHS		97 +
	49 1		98 /
	50 +		99 +
	51 g SQRT		

Usage:

Trigonometric functions:

Enter angles in degrees, -90 =< x <= 90 (*):

```

R/S => cos(x)
x<>y => sin(x)
x<>y / => tan(x)

```

Or,

```
R/S / => tan(x)
```

(*) (actual limits for signs consistency: +/- 89.99998084)

Average percent error: 0.05% (0 to 89.98 degrees).

Inverse trigonometric functions:

Enter arguments within the following ranges:

```

0 =< x < 1          for arcsin(x);
1E-50 < x <= 1      for arccos(x);
0 =< x < 1E100       for arctan(x) (**)

```

```

GT0.46 R/S => arcsin(x)
GT0.52 R/S => arccos(x)
GT0.58 R/S => arctan(x)

```

Notice that the numeric keys cross-pattern below makes remembering the entry points addresses much easier:

|8|

|4| |5| |6|

|2|

```

central key + upper key: arctan(x)
central key + lower key: arccos(x)
left key + right key: arcsin(x)

```

(**) (though ARCTAN works for negative arguments up to -1, this should not be taken into account in order to keep all functions in the first quadrant.

Average ARCTAN percent error: 0.02% within the valid range (maximum 0.06% at x=1)

Some examples:

	<i>sin</i>	<i>cos</i>	<i>tan</i>	
0.0001	0.0000 (0.0000)	1.0000 (1.0000)	0.0000 (0.0000)	(exact value at this precision)
-1.0000	-0.0175 (-0.0175)	0.9998 (0.9994)	-0.0175 (-0.0175)	
15.0000	0.2593 (0.2588)	0.9658 (0.9659)	0.2685 (0.2679)	

45.0000	0.7077 (0.7071)	0.7065 (0.7071)	1.0016 (1.0000)
89.0000	0.9998 (0.9998)	0.0175 (0.0175)	57.2041 (57.2900)
89.9900	1.0000 (1.0000)	1.7450E-4 (1.7453E-4)	5730.7974 5729.5779
89.9999	1.0000 (1.0000)	1.4120E-6 (1.7453E-6)	708193.48 572957.80
	<i>asin</i>	<i>acos</i>	<i>atan</i>
0.0000	0.0000 (0.0000)	Error 0 (90.0000)	0.0000 (0.0000)
0.0001	0.0057 (0.0057)	89.9943 (89.9943)	0.0057 (0.0057)
0.1000	5.7383 (5.7392)	84.2617 (84.2608)	5.7097 (5.7106)
0.9999	89.1899 (89.1897)	0.8101 (0.8103)	45.0239 (44.9971)
1.0000	Error 0 (90.0000)	0.0000 (0.0000)	45.0268 (45.0000)
2.0000	-	-	63.4241 (63.4349)
5.0000	-	-	78.6904 (78.6901)
999.9999	-	-	89.9427 (89.9427)
1E6	-	-	89.9999 89.9999
1E7	-	-	90.0000 90.0000

Running times are about 4 to 5.5 seconds, depending on the function (not argument dependable).

Although there are a few excellent programs for computing trigonometric functions in the HP-12C, I decided to write this one to fit some specific needs: reasonable running time, arguments and results in degrees and accurate enough for any practical purposes. The running times are not as good as I initially thought they would be by using this approach, but are reasonable. As of the accuracy, it is comparable to a good slide ruler: about 0.05%. This program uses all 99 memory steps but leaves all remaining registers free (R0 to R6 and the financial registers), however the stack is not preserved. The expressions (I) and (II) below are a fair approximation to cosine and arctangent. The dividends are hyperbolic functions close to their respective trigonometric functions and agreeing exactly at the extremes

of their valid ranges.
The divisors were obtained through curve fitting.

(I) $\cos(x) \sim (2 - \cosh(0.83840144x)) / \exp(0.15049479x^2)$, x in radians ($0.83840144 = 2 \operatorname{acosh}(2)/\pi$)

$(-\pi/2 \leq x \leq \pi/2)$

or, for programming optimization:

$\cos(x) \sim (4 - (\exp(x/68.3393) + 1/(\exp(x/68.3393)))) / (2\exp(x^2/21813))$, x in degrees,

and

(II) $\arctan(x) \sim 1.03125550 \cdot \tanh(x) / (1.03143767 - 0.032032226x^3)$, results in radians
($1.03125550 = \pi / (4 \cdot \tanh(1))$),

$(0 \leq x \leq 1)$

or

results $\arctan(x) \sim 1.03125550((\exp(x) - 1/\exp(x))/(\exp(x) + 1/\exp(x))) / (1.03143767 - 0.032032226x^3)$,
in radians

or, for programming optimization:

and $\arctan(x) \sim 1844.6((\exp(x) - 1/\exp(x))/(\exp(x) + 1/\exp(x))) / (32.2 - (\operatorname{abs}(x))^3)$; $(0 \leq x < 1)$

$\arctan(x) \sim 90 + 1844.6((\exp(-1/x) - 1/\exp(-1/x))/(\exp(-1/x) + 1/\exp(-1/x))) / (32.2 - (\operatorname{abs}(-1/x))^3)$; $x \geq 1$, results in
degrees,

The other functions are computed as follows:

for $x=0$ $\sin(x) = \operatorname{sgn}(x) \cdot \sqrt{1 - \cos^2}$ (Since $\operatorname{sgn}(x)$ is computed as $\sqrt{x^2}/x$, $\sin(x)$ fails

$\tan(x) = \sin(x) / \cos(x)$

allows it to be $\arcsin(x) = \arccos(\sqrt{1 - x^2})$ (This does not preserve the sign of the function, but

computed sequentially, as an argument to ARCCOS)

$\arccos(x) = \arctan(\sqrt{1/x^2 - 1})$ (This causes the loss of the sign, but requires less
programming steps).

Plots of (I) and (II) may show to what extent these approximations are reliable.

Edited: 7 July 2005, 9:49 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

Bad Tones / Good Tones on HP41CX

Posted by [Thomas \(CH\)](#) on 13 Jan 2005, 12:47 p.m.

Here is a complete list of good tones as tested on my HP41CX:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 15, 17, 19, 23, 25, 29, 34, 40, 42, 46, 48, 51, 52, 55, 80, 85, 86, 89, 105, 106, 109, 117, 119.

These are the bad tones, which leave the piezo buzzer in a noisy state:

10, 11, 12, 13, 14, 16, 18, 20, 21, 22, 24, 26, 27, 28, 30, 31, 32, 33, 35, 36, 37, 38, 39, 41, 43, 44, 45, 47, 49, 50, 53, 54, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 107, 108, 110, 111, 112, 113, 114, 115, 116, 118, 120, 121, 122, 123, 124, 125, 126, 127.

(The Editor may delete article nr. 454).

Kind regards, Thomas

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

GTO

[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

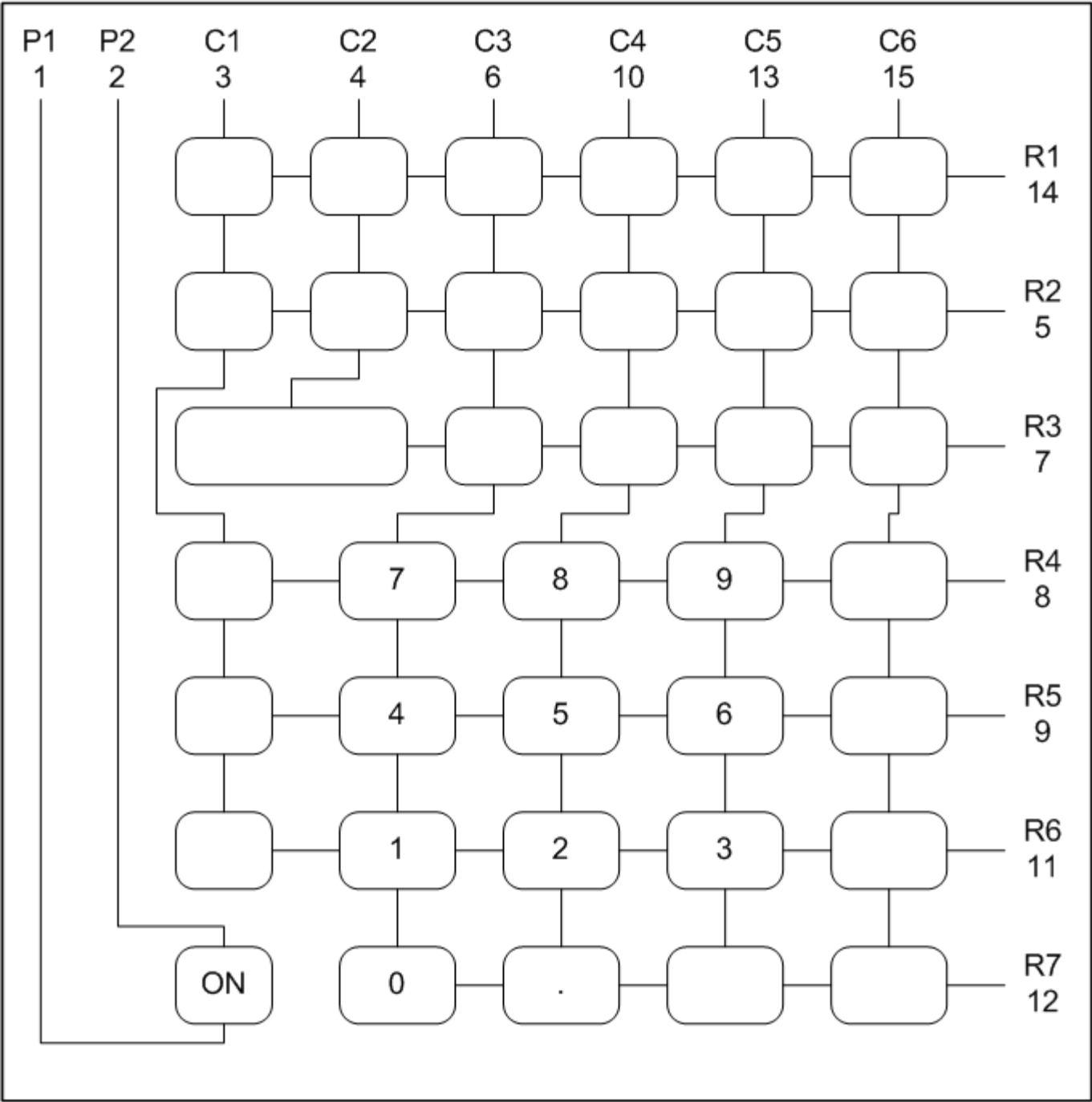
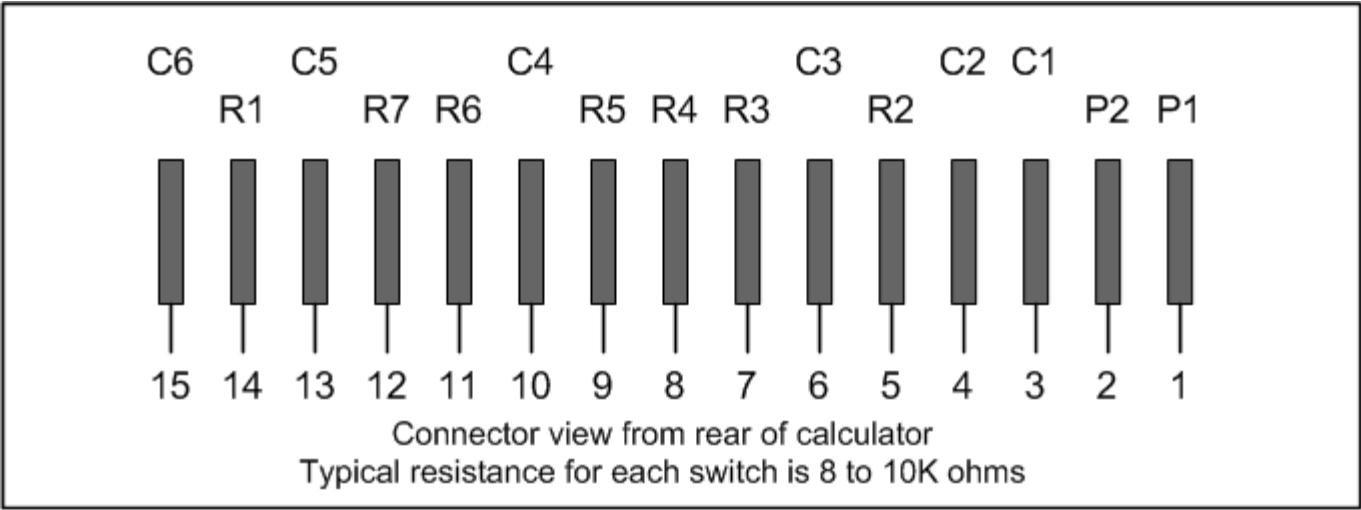
[[Previous](#) | [Next](#)]

Pioneer keyboard schematic

Posted by [Randy Sloyer](#) on 13 Nov 2004, 8:36 p.m.

All Pioneers used the same connection pinout from 1988 to 2001 with the notable exception of one version of the 32S that had a very limited production run.

Be careful probing with a meter, the carbon film at the logic board connection points is easily scraped off. Note that a closed key is somewhere in the range of 8 to 10 kohms. Any key that has open switch leakage of 10 megohms or less will cause processor lockups. The matrix view is from the front, the connector from the back.



Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Uploading ROM's for HP41 & EMU41

Posted by [Christoph Klug](#) on 24 Sept 2004, 10:35 a.m.

HP41 & EMU41 Uploading plug in module ROM-image-files

Part I : Working with ZENROM

Part II : Working with HEPAX

Part III : Generating a .BIN file

Part IV : Working with W&W RAM-BOX

Part V : Working with ERAMCO RAM-BOX

Part VI : Working with Virtual RAM-BOX

Part VII : Working with MBK-ProfiSET

Part VIII : Working with virtual MBK-ProfiSET

Appendix I : ROM file conversation HEPAX / RAM-Box

Appendix II : ROM file conversation MBK-ProfiSET / RAM-Box

Appendix III : ROM file conversation .BIN / .ROM

Appendix IV : Creating a user code software ROM for EMU41

Appendix V : Creating a user code software ROM for CLONIX

Appendix VI : Writing inside a DOS file with EMU41 screen response

Appendix VII : Storing ROM-image files on cassette tape

Appendix VIII : Back up and email transfer of ROM-image files

HP41 & EMU41 Uploading plug in module ROM-image-files

This document gives you some details about uploading ROM-images (plug in modules) from HP41 to EMU41 by using ZENROM or HEPAX or W&W RAM-Box or ERAMCO RAM-Box or MBK-ProfiSET. Two appendixes gives knowledge about ROM image file format conversion between HEPAX, MBK-ProfiSET and RAM-Box. A third appendix describes the generation of .rom files which are needed for programming CLONIX plug in modules. Last not least you get some details about creating back up, about storing ROM-image files on tape and about email transfer.

EMU41 virtual loop configuration:

First check and set the virtual loop configuration of EMU41: Open the EMU41.INI file and check / set / save the following configuration:

DISPLAY 1. Device LPRTER1 2. Device HDRIVE1 3. Device HDRIVE2 4. Device ;FDRIVE1 not used DOSLINK 5. Device XIL 700 (or 300) HP-IL/PC interface card

Wire HP41 to PC (IL-Module and HP-IL/PC interface card). Start EMU41. Switch on the real handheld and execute AUTOIO, SF 15. CCD-Module list you the devices above with CAT 0 to EMU41 screen display.

Part I : Working with ZENROM

The following parts are needed: HP41, IL-Module, Zenrom, (CCD Module), HP-IL/PC interface card, EMU41 full version. Possible HP41 port configurations :

Port 1 = ZENROM Port 2 = free Port 3 = ROM Port 4 = IL-Module

or with CCD-Module :

Port 1 = CCD Port 2 = ZENROM Port 3 = ROM Port 4 = IL-Module

If you work with CCD and Zenrom, deactivate the low letter mode of the CCD-Module as described in the CCD-Module manual (software routine TLC).

ZENROM disassembler test :

Execute 1 SELECT MANIO SF 15 and start the Zenrom MCode editor with XEQ MCED. After the COMMAND? prompt press the ENTER key for activating the disassemble mode : You get the following HP41 screen display

DIS : _ _ _ _ , _ _ _ _

Enter the start address of the page you want to disassemble, for example the lower page of port 3

DIS : C000 , _ _ _ _

Now enter the stop address for a 4Kbyte module :

DIS : C000 , CFFF

Or the stop address for a 8Kbyte module :

DIS : C000 , DFFF

Start disassembling by pressing R/S key. Now you get a test listing on EMU41 screen display. Stop listing by pressing R/S again. Now you get the COMMAND? prompt. Terminate MCED by pressing the ON key.

Writing inside a DOS text file:

Now we disassemble the module data inside EMU41OUT.DAT text file: 5 SELECT MANIO activates the DOSLINK of EMU41. Again start the disassemble mode with SF15 XEQ MCED After the COMMAND? prompt press ENTER and input start and stop address of the module :

DIS : _ _ _ _ , _ _ _ _

After pressing R/S the procedure runs, but you get no EMU41 screen response about this (refer also to Appendix IV).

Remember: Disassembling a 4KByte page or a 8KByte page takes some time. Therefore wait until transfer is finished completely and you are presented again the COMMAND? prompt. Then stop MCED by pressing the ON key. Deactivate your handheld and EMU41. Open the EMU41 folder and open the EMU41OUT.DAT file with any text editor. Store this file in same folder under NAME.LST. Open NAME.LST and edit the text :

```
MCED DIS : C000,CFFF C000 005 XROM 05 C001 00D FCNS 13 : :
```

Remove the first two leading lines, only the start address and following text lines are used. Save the file and close the editor. Now finish work with Part III.

Part II : Working with HEPAX

Also by using the HEPAX module it would be easy possible to create a disassembled ROM-image .LST file. The following parts are needed: HP41, IL-Module, HEPAX, HP-IL/PC interface card, EMU41 full version.

Port 1 = HEPAX Port 2 = free Port 3 = ROM Port 4 = IL-Module

HEPAX disassembler test :

Start EMU41, then switch on your HP41. First we disassemble to the EMU41 screen: Execute AUTOIO and start the HEPAX disassemble mode with XEQ DISASM. You get the following display message :

```
DIS : _ _ _ _
```

Enter the start address of the lower page of port 3 (= page C) :

```
DIS : C000 - _ _ _ _
```

Now HEPAX generates a new prompt for entering the stop address. For a 4Kbyte module enter:

```
DIS : C000 - CFFF
```

For a 8Kbyte module enter :

```
DIS : C000 - DFFF
```

HEPAX starts the process automatically. After looking to the screen display stop the listing by pressing R/S.

Writing inside a DOS text file :

Now we disassemble to a DOS file: Select the EMU41OUT.DAT text file by pressing 5 SELECT MANIO , this activates the DOSLINK of EMU41. Again execute XEQ DISASM and enter start and stop address :

```
DIS : C000 - CFFF
```

During the writing into the EMUOUT.DAT file you get no EMU41 screen response about this (refer also to Appendix IV). Remember: Disassembling a 4KByte page or a 8KByte will take some time. Therefore wait until transfer is finished completely. Then deactivate your handheld and EMU41. Open the EMU41 folder and open the EMU41OUT.DAT file with any text editor. Store this file in same folder under NAME.LST. Open NAME.LST and see the text :

```
C000 005 XROM 05 C001 00D FCNS 13 : :
```

There is no need to edit the HEPAX output. Close the file. Now finish the work with Part III.

Part III : Generating a .BIN file

With help of ZENROM or HEPAX we create the NAME.LST file. For generating a .BIN file we need the MS DOS prompt. Jump to the EMU41 folder with DOS command C:/ CD EMU41. Than execute the following command line :

```
clbin41 name.lst name.bin
```

Close the MS DOS window and open the EMU41 folder. Now you find the new NAME.BIN file. Size is 5,00KB for a 4Kbyte ROM-image-file or 10.00KB for a 8Kbyte ROM-image-file. Move the NAME.LST file to an other folder (for example HP41 modules) and delete the contents of the EMU41OUT.DAT file.

Installing the virtual ROM to EMU41 :

In the last step we install the new virtual ROM to EMU41. For this open the EMU41.INI file with a text editor. Add the following line :

; X 4 NAME.BIN ; comment text which includes the XROM Number

for a 4KByte ROM-image file, or

; X 8 NAME.BIN ; comment text which includes two XROM Numbers

for a 8Kbyte ROM-image-file. X is the port number and runs from 8 up to 15. By removing the leading ; you activate the virtual module for EMU41. Avoid doubling identical XROM numbers and do not place two modules on the same port address. Save the EMU41.INI file.

Start EMU41 AUTOIO (press F8 F4 AUTOIO F4) SF15 (press F5 7 15) and execute CAT 2 (press F5 N 2). Now the actual plug ins are listed. For generating .ROM files now jump to Appendix III....

Part IV : Working with W&W RAM-BOX

Alternatively to Part I (ZENROM) and Part II (HEPAX) you can use a RAM-BOX. Then the ROM-image file upload to EMU41 is much easier and faster. The following parts are needed: HP41, with W&W RAM-Box (or HP41CY), IL-Module, HP-IL/PC interface card, EMU41 full version. Possible HP41 port configurations with W&W Ram-Box or HP41CY are :

Port 1 = W&W OS / Page 9 Port 2 = Page A / Page B Port 3 = Page C / Page D & IL-Module Port 4 = Page E / Page F

Write the ROM-image file on virtual drive :

Check the first virtual drive of EMU41 by executing SF15, AUTOIO, 3 SELECT (=HPRIVE1) and DIR (= CAT 7 with CCD-Module). This lists the existing files stored on the drive to the EMU41 screen. Now enter a NAME for your ROM-image-file to the ALPHA-Register.

For writing/saving the lower Port 3 with W&W RAM-BOX you enter 12 to X-Register and execute WRTPG. Enter 13 to X-Register for writing/saving the upper Port 3 and execute WRTPG.

DIR (CAT 7) shows you the new 640 register ROM-image file on the virtual drive. Now finish the work with Part VI.

Part V : Working with ERAMCO RAM-BOX

Alternatively to Part I (ZENROM) and Part II (HEPAX) you can use a RAM-BOX. Then the ROM-image file upload to EMU41 is much easier and faster. The following parts are needed: HP41, ERAMCO RAM-Box, IL-Module, HP-IL/PC interface card, EMU41 full version. Possible HP41 port configurations with ERAMCO Ram-Box are :

Port 1 = Page +1 / Page -1 Port 2 = Page +2 / Page -2 Port 3 = Page +3 / Page -3 & IL-Module Port 4 = RSU 1 OS /

RSU 2 OS

Write the ROM-image file on virtual drive :

Check the first virtual drive of EMU41 by executing SF15, AUTOIO, 3 SELECT (=HDRIVE1) and DIR (= CAT 7 with CCD-Module). This lists the existing files stored on the drive to the EMU41 screen. Now enter a NAME for your ROM-image-file to the ALPHA-Register.

For writing/saving the lower Port 3 with ERAMCO RAM-BOX you enter +3 to X-Register and execute WRTRAM. Enter -3 to X-Register for writing/saving the upper Port 3 and execute WRTRAM.

DIR (CAT 7) shows you the new 640 register ROM-image file on the virtual drive. Now finish the work with Part VI.

Part VI : Working with Virtual RAM-BOX

For using ROM-image files, stored on virtual drive, use the advanced W&W RAM-BOX II operating system (= HP41CY), which includes two software selected 32 KByte banks = 64Kbyte RAM. Open the EMU41.INI file and activate the W&W RAM-BOX II operating system on port 8 :

8 64 ramboxii.dat ; W&W RAMBOX II XROM 31 64KByte = HP41CY

This file includes also the belonging RAM. Save the EMU41.INI file. Start EMU41, execute SF15, AUTOIO, 3 SELECT, enter the NAME of the ROM-image-file on the first virtual drive to ALPHA-Register, enter 12 to X-Register and execute READPG. Now CAT 2 lists you the imported module, and you are able to execute functions of the new virtual plug in module.

When using the virtual 64KByte RAM-Box II, you have no direct access to selected pages, by opening the EMU41.INI file. For creating .BIN (or .ROM) files work with the following and more basic EMU41 configuration : Open the EMU41.INI file and activate the 32KByte W&W RAM-BOX operating system on port 8. Activate the MLDLRAM.DAT file to port 12. Copy the MLDLRAM.DAT file, rename it to MLDLRAM.OLD and store it to the EMU41 folder :

8 4 rambox.bin ; W&W RAMBOX XROM 31 32KByte 12 8 MLDLRAM.DAT ; 8KByte RAM

Then save the EMU41.INI file. Now you have available a virtual RAM-Box with only 8KByte RAM on Port C. Start EMU41 and enter the NAME of the ROM-image-file on the virtual drive to ALPHA-Register. Enter 12 to X-Register and execute READPG. Now CAT 2 lists you the imported module, and you are able to execute functions of the new virtual plug in module.

Save the ROM-image-file as .BIN file :

Enter the EMU41 folder and rename the MLDLRAM.DAT file to NAME.BIN file. Then rename the existing MLDLRAM.OLD file to MLDLRAM.DAT file. Open the EMU41.INI file deactivate the MLDLRAM.DAT file on port 12 and add the new NAME.BIN file to the list :

; X 4 NAME.BIN ; comment text which includes the XROM Number

for a 4Kbyte ROM-image. In this case the upper 4KByte block is not used / needed. Or enter

; X 8 NAME.BIN ; comment text which includes the XROM Number

for an 8Kbyte ROM-image-file. Activate the new .BIN file on port 12 by removing the leading ; you activate the virtual module for EMU41. Avoid doubling identical XROM numbers and do not place two modules on the same port address. Last save the EMU41.INI file. Start EMU41 and work with the new virtual plug in module application.

Compared with Part I....Part III (ZENROM or HEPAX) the process of renaming the MLDLRAM.DAT file is an easy

and fast solution for creating .BIN files. For generating .ROM files now jump to Appendix III....

Part VII : Working with MBK-ProfiSET

May be you have the MBK-ProfiSET available : This RAM-BOX product, created by Mathematisches Büro Köln / Germany is housed inside an small external box, and includes a 8KByte operating system (EPROM) plus 16KByte or 32KByte RAM . For the ROM-image file upload to EMU41 the following parts are needed: HP41, MBK-ProfiSET, IL-Module, HP-IL/PC interface card, EMU41 full version. By setting the left rotary switch of the MBK-ProfiSET, you place the internal operating system to port 4 = page E + F. Insert the adapter terminal of the MBK-ProfiSET to port 1 of your HP41. Insert the IL-module to port 2 or port 3 of your HP41. Now you have available the following Block configuration :

Port 1 = Block 1 / Block 2 Port 2 = Block 3 / Block 4 Port 3 = Block 5 / Block 6 Port 4 = MBK OS / MBK OS

Write the ROM-image file on virtual drive :

Check the virtual drive of EMU41 by executing SF 15, AUTOIO, 3 SELECT (= HDRIVE1) and DIR. This lists the existing files stored on the first virtual drive to the EMU41 screen. Now enter a NAME for your ROM-image-file to the ALPHA-Register.

For writing/saving Block 3 with the MBK operating system, load the Block number to X-Register and execute SLCT. Enter the file name to ALPHA-Register and execute SAVEBL or GETBL. DIR lists you the new 640 register ROM-image file on the virtual drive. Now finish the work with the following Part VIII :

Part VIII : Working with virtual MBK-ProfiSET

Open the EMU41.INI file and activate the MBK-ProfiSET operating system on port 8. Activate the MLDLRAM.DAT file to port 10. Copy the MLDLRAM.DAT file, rename it to MLDLRAM.OLD and store it to the EMU41 folder :

8 8 mbkprof.bin ; MBK PROFISET XROM 27 + XROM 31 version 2G 0 8 MLDLRAM.DAT ; 8KByte RAM

Then save the EMU41.INI file. Now your have available a virtual MBK-ProfiSET with small 8KByte RAM on page 10 (= Block 3) and page 11 (=Block 4). Start EMU41 and enter the NAME of the ROM-image-file on the virtual drive to ALPHA-Register. Execute 3 SLCT followed by GETBL. Now CAT 2 lists you the imported 4KByte module, and you are able to execute functions of the new virtual plug in module.

Appendix I : ROM file conversation HEPAX / RAM-Box

The RAM-Box ROM file format of W&W and ERAMCO is identical and also compatible to the ZEPROM file format (command ILBURN). Unfortunately this format is not compatible to the HEPAX ROM-file format.

With help of EMU41 it is possible to convert ROM-files between the RAM-Box and HEPAX format. Install the W&W RAM-Box operating system and the HEPAX operating system to EMU41 by editing EMU41.INI file :

8 16 hepax1d.bin ; hepax XROM 07 bank switched module 11 4 rambox.bin ; rambox w&w XROM 31

When starting EMU41, HEPAX places the belonging HEPAXRAM.DAT file automatically to port 8 and port 9, the operating system is placed to port 10. It is possible to use the HEPAX RAM on port 8 and port 9 also with the W&W RAM-Box operating system.

Therefore you are able to transfer ROM-files between page 8 page 9 and the virtual drive of EMU41 by executing the HEPAX commands WRTROM or READROM or the W&W RAM-Box commands WRTPG and READPG. Place the filename to Alpha-Register and the port number (in this configuration 8 or 9) to the X-Register. Than execute one of the four commands.

Last you transfer the ROM-files from the virtual drive to your HP41 handheld with RAM-Box or HEPAX.

Appendix II : ROM file conversation MBK-ProfiSET / RAM-Box

Like Appendix I, the MBK-ProfiSET ROM file format is different from the common W&W and ERAMCO file format, which is also compatible to the ZEPROM file format (command ILBURN).

With help of EMU41 it is possible to convert ROM-files between MBK-ProfiSET and RAM-Box format. Install the W&W RAM-Box operating system, the MBK-ProfiSET operating system and the MLDLRAM.DAT file to EMU41 by editing EMU41.INI file :

8 4 rambox.bin ; rambox w&w XROM 31 14 8 mbkprof.bin ; MBK PROFISET XROM 27 + XROM 31 version 2G 10 8 MLDLRAM.DAT ; 8KByte RAM

Now you are able to transfer ROM-files between page 10 (= Block 3), page 11 (=Block 4) and the virtual drive of EMU41 by executing the W&W RAM-Box commands WRTPG or READPG and the MBK-ProfiSET commands SLCT plus SAVEBL or GETBL : Place the filename to ALPHA-Register and the port or Block number to the X-Register. Than execute the commands listed above.

Last you transfer the ROM-files from the virtual drive to your HP41 handheld and use them with your real RAM-Box or MBK-ProfiSET.

Appendix III : ROM file conversation .BIN / .ROM

The former version of EMU41 from Jean-Francois Garnier works with ROM-image files stored in the .bin format. V41 from Warren Furlow works with the different .rom format. TOP (the other page) contains some .rom files for downloading. Actual Jean-Francois Garnier added the .rom format to EMU41, and you are able to use both formats in parallel. Now EMU41 includes the updated CLBIN41.EXE software tool (version V1.8 JFG 2004) for converting 4KByte .bin files to .rom files :

clbin41 name.lst name.bin = convert .lst file into .bin file
clbin41 null name.bin = creates empty .bin file
clbin41 /r name.rom name.bin = convert unpacked .rom file into packed .bin file
clbin41 /b name.bin name.rom = convert packed .bin file into unpacked .rom file
clbin41 /? = view help text and software version

.bin files need less memory space and include 4KByte or 8KByte ROM-images and advanced bank switched ROM applications. .rom files need more memory space and contain only 4KByte ROM-images. For programming CLONIX you also need the .rom format...

Appendix IV : Creating a user code software ROM for EMU41

May be you want saving main memory by storing your own user code program applications to a 4KByte ROM-image page. For creating this, EMU41 plus virtual RAM-Box are the needed software tools : Transfer your user code program solutions from HP41 to the virtual drive of EMU41 (AUTOIO, 3 SELECT, WRTP) and load them to EMU41 main memory (AUTOIO, 3 SELECT, READP). Next install the virtual RAM-BOX by editing the EMU41.INI file :

8 4 rambox.bin ; RAMBOX W&W XROM 31 10 8 MLDLRAM.DAT ; 8KByte RAM

Working with RAM-Box : First initialise 4KByte page 10 with XROM number. A list of XROM numbers you find in the EMU41.INI file of the author. Avoid doubling XROM numbers with your configuration. Enter a page name to ALHPA-Register. This page name is the module name and listed with CAT 2. Load the XROM number to Y-Register and page number 4 to X-Register. Now execute INITPG.

Next load your user code programs from main memory to page 10. Enter the program name in the ALPHA-Register and page number 10 in X-Register. Execute LDPGM. The FAT of your 4KByte module application accept maximally 64 commands = global ALHPA LBL's. Therefore replace not needed global LBL's of your software routines by numerical LBL's before uploading to RAM-Box. Next enter the program name in ALPHA-Register and execute

XQ>XR. This transforms the XEQ commands and jump distances to an fast speed XROM command. Remove the uploaded programmes from main memory.

List your virtual plug in module application with CAT 2 and execute some software tests inside EMU41. If anything works fine, calculate the check sum of your module application : Enter the page number 10 to X-Register and execute PGSUM. Close the page by entering the page number 10 to X-register, a four letter ROM version to ALPHA-Register and execute ENDPG.

Now your virtual user code software module is finished. Copy the MLDLRAM.DAT file and rename it to NAME.BIN for former use with EMU41.

Appendix V : Creating a user code software ROM for CLONIX

The new CLONIX plug in module from Diego Diaz is a fair price solution for HP41 users, who do not own a RAM-Box, HEPAX or Zeprom. CLONIX includes 6 x 4KByte ROM pages. Below you find an advanced HP41 port configuration, the CLONIX contains 6 ROM-image files, programmed from pages 8 to page 13.

Port 1 = CLONIX Port 2 = DOUBLE X-MEMORY Port 3 = IL-Module Port 4 = free

For programming a CLONIX, you need a PC (WIN98 or WIN2000/XP) and the special programming adapter (interfaced to the RS232 COM port) and software from Diego Diaz. His detailed user manual describes the programming procedure including some screen displays. By loading/programming ROM-image files to CLONIX now you also get high priced or exotic plug in solutions like CCD, ZENROM, PPC, ADVANCED, EXT-I/O, EXT-IL, SKWIDINK.... Some of this .rom files you find on TOP (the other page) of Warren Furlow. The conversion of existing 4KByte .bin files to .rom files (needed for programming CLONIX) is described in Appendix III.

May be you want saving main memory of your HP41 by storing your own user code program applications to a CLONIX page. Than you need a special .rom file containing your own software application. For creating this, EMU41 plus virtual RAM-Box and virtual Zenrom are the needed software tools. Start with Appendix IV, as result you get a MLDLRAM.DAT file on port 10 containing your virtual module application. Next install the virtual Zenrom to EMU41 by editing the EMU41.INI folder :

8 4 rambox.bin ; RAMBOX W&W XROM 31 9 4 zenrom.bin ; Zenrom XROM 05 10 8 MLDLRAM.DAT ; 8KByte RAM

Up to now only 8KByte MLDLRAM.DAT files exist. We are able to rename this to a 8KByte .bin file – but it is not possible to convert this to a .rom file (Appendix III) because .rom files limited to 4 KByte pages. May be we get a 4KByte MLDLRAM.DAT file with an future EMU41 update, actual we use the following workaround with help of virtual Zenrom :

We disassemble the 4KByte module on page 10 to the EMU41OUT.DAT text file: 5 SELECT MANIO activates the DOSLINK of EMU41. Now start the Zenrom disassemble mode with SF15 XEQ MCED After the COMMAND? prompt press ENTER and input start and stop address of page 10 :

DIS : _ _ _ _ , _ _ _ _

Enter the start address of page 10 :

DIS : A000 , _ _ _ _

Enter the stop address of page 10 :

DIS : A000 , AFFF

After pressing R/S the procedure runs, but you get no EMU41 screen response about this (refer also to Appendix IV).

Remember: Disassembling a 4KByte page takes some time. Do not use the fast speed mode of EMU41. Instead wait until transfer is finished completely and you are presented again the COMMAND? prompt. Then stop MCED by pressing the ON key. Deactivat EMU41. Open the EMU41 folder and open the EMU41OUT.DAT file with a text editor. Store this file in same folder under NAME.LST. Open NAME.LST and edit the text :

```
MCED DIS : C000,CFFF A000 00X XROM 0X A001 00X FCNS XX
```

Remove the first two leading lines, only the start address and following text lines are used. Save the file and close the editor. Next we generating a .BIN file, for this we need the MS DOS prompt. Jump to the EMU41 folder with DOS command C:/ CD EMU41. Than execute the following command line :

```
clbin41 name.lst name.bin
```

Close the MS DOS window and open the EMU41 folder. Now you find the new NAME.BIN file. Size is 5,00KB for a 4Kbyte ROM-image-file. Move the NAME.LST file to an other folder (for example HP41 modules) and delete the contents of the EMU41OUT.DAT file. Next we convert the name.bin. file to a .rom file by using the MS DOS prompt command line :

```
clbin41 /b name.bin name.rom
```

Close the MS DOS window and open the EMU41 folder. Now you find the new NAME.ROM file. Size is 8,00KB for a 4Kbyte ROM-image-file. Transfer the .rom file to the CLONIX programming tool folder and start programming procedure as described in the user manual of Diego Diaz.

Appendix VI : Writing inside a DOS file with EMU41 screen response

May be you prefer to get a real-time screen response during uploading great files (like a disassembled ROM-image-page) inside a DOS text file, as we know this from TRANS41 software.

As Jean-Francois Garnier told me, this is also possible with EMU41 : At DOS prompt start EMU41 with the /lg extension (= logger). Than EMU41 automatically generates a DISP.LOG file and transfers the screen output also to this file. When the data upload is completed leave EMU41 (press F1) and open the DISP.LOG file which contains the data.

Appendix VII : Storing ROM-image files on cassette tape

If you want to store ROM-image files with RAM-BOX on tape (IL-Cass-Drive) execute NEWM 027. After writing 12 ROM-image files on tape add a 300 register data file with CREATE, named ENDTAPE. Now you are able to store 12 more ROM-image-files on the second track of the tape.

Appendix VIII : Back up and email transfer of ROM-image files

The actual version of EMU41 now includes two virtual mass storage drives HDRIVE1 and HDRIVE2. For creating a back up copy from your real tape or disk use the EXT-I/O commands MCOPY or COPYFL. Now this advanced functions are also run correctly with the virtual drives of EMU41.

For easy and fast exchange of HP41 ROM-image files by email copy the HDRIVE1.DAT or the HDRIVE2.DAT file from the EMU41 folder (= virtual tape or disk) and transfer them to other HP41 enthusiasts.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Primitive RPN Calculators

Posted by [Palmer O. Hanson, Jr.](#) on 21 Apr 2004, 12:34 a.m.

When one thinks of calculators which use the RPN system one usually thinks of the Hewlett-Packard product line. National Semiconductor Corporation (NSC) also made a number of calculators using the RPN system. Two of the NSC RPN devices, the NSC Model 600 and the Novus 650 Mathbox, are easily the most primitive electronic calculators in my collection. NSC made more versatile RPN calculators as well. Two that are in my collection are the NOVUS 4510 Mathematician and the NS4615 Programmable. Other RPN calculators from NSC include the NS 900, the 4515 Mathematician PR and the NS 4525 Scientist.

The NSC Model 600 and Novus 650 Mathbox are four function calculators which implement RPN. The devices have only 15 keys: ten keys for numbers 0 through 9, four keys for the four functions and a clear key. There is no ENTER key. The Enter function and + function are combined on a single key. The devices do not have a decimal point key because they only operate with integers. If the user divides 2 by 7 the machine returns zero in the display. The instruction booklet explains that if you want to see fractional parts you must add trailing zeroes to the numerator and select the position of the decimal point for yourself. That is very similar to the situation we encountered with slide rules in the days before the advent of hand-held electronic calculators. Both devices display a decimal point between the second and third digits from the right as an aid to those who add and subtract dollars and cents but the decimal point has no effect on the numerical display. Calculators can't get much more primitive than that.

The NOVUS 4510 Mathematician which came on the market in 1975 offers many of the capabilities of the HP-35. The exceptions were:

- It uses eight digit data registers where the mantissa is truncated not rounded.
- It does not offer scientific notation (EE).
- It does not offer true roll-down of the stack . Pressing the clear key (C) brings the contents of the y register to the x register and enters a zero into the z register.

The Mathematician has a second function key (F) which allows it to offer several capabilities not available with the HP-35 including:

- Degree to Radian and Radian to Degree conversion.
- Addition and subtraction into the memory register.
- An $M + x^2$ function.

The $(M + x^2)$ function does not change the contents of the X register but squares the contents of the x register and adds the result to the memory. This feature permits efficient accumulation of the sums needed for single variable statistical analysis. To do so simply clear the stack and the memory, enter each x value in order and press F ($M + x^2$) +

for each entry. The sequence ends with sum x in the display and sum x in the memory, When the data entry is complete press ENT ENT, enter the number of entries, press / and see the mean in the display. Press x CHS MR +, enter the number of entries -1, press / and SQRT and see the standard deviation in the display.

Let's try to do that with the HP-35. To accumulate the sums clear the stack and the memory, then enter each x value in order and press ENTER ENTER x RCL + STO Rdown + for each entry. The sequence ends with sum x in the display and sum x^2 in the memory. When the data entry is complete press ENTER ENTER , enter the number of entries, press / and see the mean in the display. Press x CHS RCL + , enter the number of entries -1, press / and SQRT and see the standard deviation in the display.

The sums could also be generated with two passes through the set of input values but it should be clear how much more efficient the process is if one has the $(M + x^2)$ function at one's disposal. Of course, once one moves to calculators such as the HP-45, TI-30, etc., which have summation functions then the process becomes easier and the machine counts the number of entries as well. One needs to evaluate the advantages of the Mathematician in the context of the time at which it was used.

The NS4615 Programmable offers the same features as the NOVUS 4510 plus a 102 step programming capability. The versatility of the programs which can be written is limited by the absence of conditional branches. I have not been able to demonstrate the programming capability with the device in my collection which has some internal damage due to leakage from the hard-wired rechargeable batteries.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

"Linear Programming" to Minimize LBL Usage on the HP-33s

Posted by [Paul Brogger](#) on 15 Mar 2004, 8:55 p.m.

The HP-33s, having 31K memory available but only 26 labels and 27 registers, will challenge us to program in different ways than we may have in the past.

One idea for minimizing label usage is to utilize Flags as either parameters to routines, or to encode parameters passed through other means. The routine below provides convenient keyboard STO/RCL access to the statistics registers, and illustrates encoding the sign of the first argument as a STO (positive) or RCL (negative) flag. In this manner, two related functions may be combined under a single LBL code.

Of course, another method of minimizing LBL use is unwinding subroutines (repetitively coding subroutine logic in-line), but this simple program wouldn't likely have embodied any subroutine calls. A variation on the approach used in this routine enabled me to write STO & RCL functions for any of 10 3-dimensional vectors (numbered 1-10), again under a single label. (That one does have some would-be subroutine logic strung out three times in the program body.)

(The program also shows how comments may be entered as an equation after an "FS? 10" test.)

I hope you find it useful. Comments or corrections are welcome: take the "NoHormel" out of my email address & fire away!

STO / RCL Sum (stat) registers

Accepts: 1 <= |x| <= 6 to specify register
if x negative: RCL, otherwise STO

STO stores value saved in original y

Index mapping is |x|+27 -> i .

```
E0001 LBL E
E0002 FS? 10
E0003 STO+RCL: STAT REGS
```

```
E0004 RUP      move t to x
E0005 STO i    save original t for later
E0006 RDN      restore stack
```

```
E0007 CF 4     clear Flag 4 ("STO" flag)
E0008 x>0?     if x not negative
E0009 SF 4     then "STO" is indicated
```

```
E0010 ABS      remove sign
E0011 6         six is upper limit for this function
E0012 x<y?     if original x is greater
```



```
E0013  x<>y      then substitute 6
E0014  RDN      and use whatever is now in y

E0015  1        one is lower limit for this function
E0016  x>y?     if original x is lesser
E0017  x<>y     then substitute 1
E0018  RDN      and use whatever is now in y

E0019  27      bias index by 27
E0020  +        to obtain true stat register index

E0021  RCL  i   restore saved original t
E0022  RDN      and put back to simulate stack drop

E0023  x<>i     swap adjusted stat register index with i
E0024  RDN      and move original y to x

E0025  FS?  4   was original x positive (ST0)?
E0026  STO  (i) if so, store original y in indexed var.

E0027  FS?  4   is this a STO call?
E0028  RDN      if so, get rid of original y

E0029  RCL  (i) whether STO or RCL, read addressed var.

E0030  CF  4    clear "positive" flag
E0031  RTN
```

CK=163D Note: Checksum and length values will depend upon:
LN=115 a. choice of "E" for program LBL,
 b. equations used to enter constants, and
 c. exact text of equation comment at E0003.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-like RPN Java calculator for mobile phones

Posted by [Roar Lauritzsen](#) on 27 Feb 2004, 4:16 a.m.

I was frustrated by the simplistic built-in calculators on most mobile phones, but I realized that the power was in my own hands when I got myself a Java programmable phone, so I set about to make a scientific RPN calculator application.

After much hard work in my spare time, I'm reasonably pleased with the result, and apparently many users are enthusiastic about it. One of them wrote the following to me in an e-mail: "tears almost welled in my eyes out of sheer joy and excitement. I played with the calculator until around 2am in the night".

Well, it may not be *that* good, but you can take a look here:



<http://gridbug.ods.org/Calc.html>

(Be patient, gridbug.ods.org is hosted with only 128 kbps upload capacity, and the quality-of-service of my ISP is terrible, so if you cannot connect, try again in 1/2 hour)

Best regards,
Roar Lauritzsen

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

homebrew woodstock batt pack

Posted by [kevin clevenger](#) on 27 Feb 2004, 12:49 a.m.

just wanted to give back a little after ive taken so much :D

noticed that there was a classic battery back from scratch article but didnt see anything listed for a woodstock submitted for your approval

http://66.87.77.56:5150/hp/hp_batt_pack.html

kev.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP12C Platinum "curiosities"

Posted by [Vieira, Luiz C. \(Brazil\)](#) on 24 Jan 2004, 9:01 p.m.

I read the HP12C Platinum Owner's Handbook & Problem Solving Guide (OHB&PSG) and I found some conflicting information regarding some facts exposed below. In fact, I counted on Gene Wright and Tony Hutchins who helped me when tracking down some of these conflicts, and I wish to thank them both for that. Maybe there are yesterday news here, so I ask you to forgive me if I'm talking about facts that are already known.

Indirect-Access-only Registers

This is not exactly a conflicting information; in fact, I saw no mention to these facts in the chapter I think they should be mentioned in the HP12C Platinum OHB&PSG. Please, if you have found something about this in the HP12C Platinum documentation, let me know.

As mentioned by Katie Wasserman in her excellent [HP12C Sorting](#) article and by Valentin Albillo in its "Serendipitous Solver" (follow the thread pointed above), all available numbered registers + [FV] (from the TVM register set) can be indirectly accessed by using [n]-contents as index. This gives us the "regular" HP12C the ability to hold up to 21 values accessed indirectly.

The HP12C Platinum allows up to 31 "cash flow data" to be stored and manipulated. The fact is that the first 20 values can be directly accessed the same way in both HP12C and HP12C Platinum. If the HP12C Platinum memory is cleared after any specific procedure OR no steps after step # 239 have yet been used, CF20 to CF29 exist and can only be accessed with [g][CFj] and [RCL][g][CFj], provided that [n] has the correct "indexing" value (from 20 to 29). Also, [PV] can be accessed with 30 as index. Keep in mind that in both calculators [g][CFj] always increase [n] by 1 BEFORE copying X-register contents to the indexed register, and [RCL][g][CFj] decreases [n] by 1 AFTER retrieving the indexed register contents back to X-register and also that [PV] holds the last possible cash flow data.

BTW, in both calculators the [Nj] register set seems to be composed by a group of single unsigned 1-byte, BCD-coded registers. Their range goes from 1 to 99, no negative, no fractional parts.

"Weird" Memory Count

This is something I'd like to read more about and know from your experience.

Those who have already programmed in the HP12C know the "memory shared" space, showed below...

CF reg.	# reg	PRGM steps	
CF0	R0		
CF1	R1		
.	.		
.	.		
CF6	R6		
CF7	R7	99 to 93	<- always available

CF8	R8	92 to 86
.	.	.
CF19	R.9	15 to 09
CF20	FV	not applicable

I tested the HP12C Platinum memory usage after keying in program steps at the first byte of each available register, that was actually converted to seven program steps (bytes) automatically after keying them in. If we consider that all memory is available (memory cleared) we have:

CF reg.	# reg	PRGM steps
CF0	R0	
CF1	R1	
.	.	.
CF6	R6	
CF7	R7	399 to 394
CF8	R8	393 to 387
.	.	.
CF19	R.9	316 to 310
CF20	---	309 to 303
CF21	---	302 to 296
.	.	.
CF29	---	246 to 240
CF30	FV	not applicable

<- always available
(weird: six bytes only ????)

The weirdness is related to CF7. I performed the "memory filling" maneuver at least two times to conclude this fact. Has anyone else checked for this? Is it correct? We all know that there is no way to "produce" a non-existing byte if it is not available. Otherwise it's possible to hide it, so the system uses it (R7) but the user doesn't "see" it (steps 394 to 399). If there is a 400 step # everything is fine, but...

[f][RPN] & [f][ALG]

These are actually the major "enhancement" offered with the new Platinum. Of course, as a consequence of the additional Algebraic mode, many "new" features are also available. The fact is that in Beta, pre-production units, these keystrokes could not be part of a program, i.e., are "non-programmable" sequences. The production-line units accept these key sequences as part of a program. Based on Beta-units behavior, these features were actually supposed to be not-programmable, and all program examples shown in the Platinum OHB&PSG consider them this way. Maybe the OHB&PSG original text was prepared with Beta units in mind, and these keys were not programmable in them. As for the examples, [f][RPN] and [f][ALG] are supposed to be pressed after PRGM mode is active and not generate keycodes or change program steps. In actual, regular Platinum units, pressing any of these key-sequences will add a program step with related keycode (42 16 for [f][RPN] and 42 26 for [f][ALG]).

No Return to zero after [f]CLEAR[PRGM] in RUN mode

This is completely based on Tony Hutchins very observation; his are the credits. And Tony adds that he knew about this from Jordi Hidalgo's Datafile article. I was not aware about this fact before Tony's mentioning in one of his e-mails. Credits to those who deserve!

If we press [f][P/R] twice, in sequence, the program "pointer" is always reset to 000, but [f]CLEAR[PRGM] does not do that (in the regular HP12C it does). Consider you have an HP12C Platinum with all memory cleared (you don't need to clear the memory contents of your calculator, just take care not causing an unexpected [Error *] message). In RUN mode, press:

[g][GT0]005

It always work because step #005 is always available. If you press [f][P/R] you'll see

[005, 43, 33, 000]

or

[005, anything_else]

if you have programs stored in your calculator. Now, press [f][P/R] twice and you'll see

[000,]

Now get back to RUN mode and try this:

[g][GTO]005
[f]CLEAR[PRGM]
[SST] (hold it for a while)

In a regular HP12C you'd see:

[01-43,33 00]

But in the HP12C Platinum you'll see:

[005,43,33,000]

(Remember we're assuming you have no programs stored in the calculator.) That's it! Program pointer was not set to [000,] as expected. And the "reset to 000 after [f]CLEAR[PRGM]" is mentioned in the HP12C OHB&PSG, although it does not work. You'd need to press [g][GTO]000 instead of [f]CLEAR[PRGM].

There's more about the HP12C Platinum, but I think there's something very important afterall: it's a curious, interesting calculator. Even if HP/Kinpo guys did not intend to, it has some "hidden" mysteries that, at least, are "teasing".

Please, if you have any suggestions, comments or want something to be added here, let me know.

Luiz C. Vieira - Brazil

lcvieira@quantica.com.br

Edited: 21 Mar 2004, 1:33 a.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-49G+ -- A Look Inside . . .

Posted by [Paul Brogger](#) on 8 Dec 2003, 9:50 p.m.

I was able to get my HP-49G+ apart and take a few pictures. I thought I'd share them with you all, and Dave has been kind enough to host them for us.

The [first one](#) shows the two halves not quite fully separated, and the IR lens removed.

The upper, dark section is the inside of the bottom case half. The battery compartment is to the left, and the wires connect the AAA and backup batteries and a large electrolytic capacitor to the main board. The small silvery rectangle near the top is a piece of conductive foam serving as the negative battery contact with the PCB. The larger black circle is where the buzzer is mounted, the smaller concentric circles to its upper left are the reset switch (a springy rubber contact).

The lower half shows the front case half, with the LCD board on the left covered by self-stick foil-backed paper. (It's actually aluminum-colored -- it looks coppery in these pictures, for some reason.) The main (CPU/key/SD) board is on the right.

The IR lens is thin and flexible, and pops out easily, but would probably be hard to get back in without taking apart the case halves (at least partially).

The [second picture](#) shows the electronics a bit more closely. A capacitor is glued to the main board with a smear of red-orange goo, and the reset contact is some gold fingers interleaved near the blue jumper wire.

The main board connects to the LCD board via: 1) a mylar (?) ribbon that runs across the calculator, just to the right of the coppery-looking shielding, and 2) a series of solder connections along two edges of what looks like a rectangular extension of the LCD board, near the bottom of the picture.

The square black chip above & to the right is marked "ARM" -- the other two I haven't looked at too closely. There's another pad of conductive foam attached to the SD card slot assembly.

The [third image](#) shows most of the self-stick paper/foil backing peeled away from the LCD board, exposing two blobs of black stuff apparently sealing LCD controller chips on the board.

Also visible are the IR sender & receiver, and the USB mini-B socket.

This view gives a good idea how the thing is put together -- the two posts are what I drilled to free up the top end, and otherwise it's only a series of plastic tabs around the edge that lock into the bottom case. After "topping" the posts, the two halves snapped apart quite easily.

BTW, the black accent that surrounds the calculator face is actually applied, and the calculator's gold-colored front and "side bumper" are actually one piece. Don't pry upward on the black accent, as it's just decoration and won't get you anywhere.

And, in case you're wondering, this unit is not dead -- it survived the "operation" fine. One key is that I didn't try to pull the self-stick aluminized paper completely off, as it's in contact with the LCD edge of the mylar ribbon connecting the two circuit boards. I think that, were I to try to take all of the sticky paper/shielding off, the ribbon connector may well come with it

Addendum: More Images

49G+ Chips Close-up

In response to a request, I took and posted a [close-up of the chips](#), and someone chimed in with an [explanation](#).

49G+ Battery Compartment Close-ups

In response to a thread in comp.sys.hp48 about concern over stretching springs in the battery compartment to fix battery contact problems, I took several close-ups of the compartment from the inside.

The first three were taken with the cellophane tape and plastic insulating sheet in place:

1. [battery compartment close-up 1](#)
2. [battery compartment close-up 2](#)
3. [battery compartment close-up 3](#)

The fourth was taken with the cellophane tape removed to expose the wires more clearly:

4. [battery compartment close-up 4](#)

The last was taken with the black plastic insulator/shield sheet folded toward the camera, to expose the back-up battery and its connections:

5. [battery compartment close-up 5](#)

49G+ Keyboard Close-ups

Perhaps the most talked-about issue surrounding the introduction of the 49G+ has been early problems with its keyboard. I felt that the disassembly would not be complete without a close look at how its keyboard was actually put together:

1. [PCB removal](#)
2. [key parts \(front\)](#)
3. [key parts \(back\)](#)
4. [keyboard sandwich on PCB](#)
5. [sandwich/LCD close-up](#)

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Fifteen Digit Precision on the HP-32S

Posted by [Palmer O. Hanson, Jr.](#) on 18 Oct 2003, 12:43 a.m.

FIFTEEN DIGIT PRECISION ON THE HP-32S

We start with some arithmetic exercises using the statistical functions of the HP-32S:

Consider the numbers $x = 1.000007$

$y = 1.000004$

$x^2 = 1.000014000049$

$y^2 = 1.000008000016$

$xy = 1.000011000028$

where the squares and the products have thirteen significant digits. In calculators such as the HP-32S the squares and the product are rounded to twelve significant digits and displayed as 1.00001400005, 1.00000800002 and 1.00001100003. The rounding process essentially adds the values of 1E-12, 4E-12 and 2E-12 to the exact values to get the rounded values. Perform the following sequence from the keyboard of an HP-32S:

CLSUM clears the statistical registers 1.000004 ENTER 1.000007 SUM- LAST x SUM+

Use the STAT function to view the summation registers and see

SUMx = 0 SUMy = 0 SUMx² = -1.E-12 SUMy² = -4.E-12 SUMxy = -2.E-12

where the sums in x², y² and xy contain the negatives of the values which are added during rounding of the squares and product. Let's try another arithmetic exercise using the statistical functions:

Consider the numbers $x = 111,111$

$y = 1,111,111,111$

$x^2 = 12,345,654,321$

$y^2 = 1,234,567,900,987,654,321$

$xy = 123,456,666,654,321$

where in the HP-32S the squares and the product become

$x^2 = 12,345,654,321$

$y^2 = 1.23456790099E18$ displayed as $1.2345679E18$

$xy = 1.23456666654E14$ displayed as $1.23456667E14$

To see the actual twelve digit mantissas you can use the SHOW function. Perform the following sequence from the keyboard:

CLSUM clears the statistical registers 1,111,111,111 ENTER 1,111,111,111 to y 111,111 111,111 to x SUM-

where we expect the negatives of the rounded values to appear in those registers and do; for example for the square y^2 , perform the sequence

STAT SUM y^2

and see $-1.2345679E18$ in the display with a mantissa of -123456790099 with the SHOW function. For the product xy , perform the the sequence

STAT SUM xy

and see $-1.23456667E14$ in the display and a mantissa of -123456666654 with the SHOW function.

Now, perform the following sequence which would be expected to remove the entered values from the statistical registers such that the values for SUMx, SUMy, SUM x^2 , SUM y^2 and SUM xy should be zeroes:

1,111,111,111 ENTER 111,111 SUM+

SUM $x^2 = 0$, but

SUM $y^2 = -2,350,000$

SUM $xy = 321$

The result in SUM y^2 can be obtained by subtracting a value for y^2 which uses a rounded twelve digit mantissa from a value which uses a truncated (not rounded) fifteen digit mantissa; that is,

1,234,567,900,987,650,000 the fifteen digit truncated value of y^2 - 1,234,567,900,990,000,000 the twelve digit rounded value of $y^2 = -2,350,000$

The result in SUM xy can be obtained by subtracting a value for xy which uses a rounded twelve digit mantissa from the actual fifteen digit value of xy ; that is,

123,456,666,654,321 the fifteen digit value of xy - 123,456,666,654,000 the twelve digit rounded value of $xy = 321$

I obtain the same results with my HP-10B. These results are compatible with what was described as "... a tricky property of the SUM- and SUM+ keys whereby certain calculations can be carried out to 13 significant digits before being rounded back to 10 ..." in a footnote on page 208 of the HP-15C Advanced Functions Handbook, but adjusted for the 12 digit mantissas in the HP-32S and HP-10B. I do not get similar effects with older HP calculators such as the HP-33C, HP-67 and HP-80.

Pages 205-211 of the HP-15C Advanced Functions Handbook proposes eight cases for evaluating the ability of a quadratic program to solve difficult cases. It presents a single precision program for the HP-15C which can obtain correct results for only three of the eight cases. It also presents a program which uses the "tricky properties" in a double precision routine which carries twenty digits during the calculation of the discriminant. That program obtains correct results for all eight cases. It uses eleven data registers and is quite slow. An equivalent program on my HP-41C runs for as much as ten seconds to obtain a solution. In 1991 I wrote a single precision program for the TI-59 which uses truncated thirteen digit mantissas as opposed to the rounded ten digit mantissas used with the HP-11, HP-15 and

HP-41. My TI-59 program obtained correct results for seven of the eight cases. It failed for the seventh case where the discriminant is the single digit difference between two twenty digit numbers. That program and similar programs for the TI-74, TI-81 and Casio fx-7000G are described on pages 8-13 of the Volume 14 Number 3 issue of TI PPC Notes.

The four cases which a single precision TI-59 program solves correctly but a single precision HP-41C program solves incorrectly (cases 4,5,6 and 8) all involve discriminants which are calculated as the difference between two numbers of 11 to 13 significant digits. Using the insight into the "tricky properties" of the SUM+ and SUM- functions derived from the numerical exercises similar to those above I wrote an HP-41 program which carries 13 significant digits during the calculation of the discriminant. That program yielded correct solutions to seven of the eight cases, used fewer registers and used less run time than the double precision program. The HP-41 program is documented in a 23 July 2003 entry in Archive 13 of the HP Forum with the title "Update: 13 Digit Precision on the HP-15C".

A similar program for the HP-32S follows which will solve quadratics with discriminants which are calculated as the difference between two numbers of up to 15 significant digits. It follows the convention in the HP-15C Advanced Functions Handbook where the quadratic is defined as $c - 2bx + ax^2 = 0$ and the discriminant is calculated as $d = b^2 - ac$. That definition reduces the size of the two parts of the discriminant by a factor of four. The program calculates the terms of the discriminant in stages. First, the 3 least significant digits of the ac term are accumulated in statistics register SUMxy using the "tricky properties". That value is recalled and stored in the stack. Then the 3 least significant digits of the b^2 term are accumulated in statistics register SUMy^2 using the "tricky properties". Finally, the 10 most significant digits of b^2 and ac are calculated in the normal manner and combined before the smaller corrections for the least significant parts are applied. The program uses five data registers A through E. The user solves by entering the constants of the quadratic in response to prompts from the INPUT command and running the program. For real roots the program stops with one root in the display. The second real root is displayed by pressing $x \leftrightarrow y$. If the solution is complex Flag 0 is set. The program stops with the real part in the display. The imaginary root is displayed by pressing $x \leftrightarrow y$ where + and - signs for the imaginary part are implied. The run time is less than one second. The program listing is:

```
Q01 LBL Q Q02 CF 0 Q03 INPUT C Q04 INPUT B Q05 INPUT A Q06 CLSUM Q07 RCL C Q08 SUM+ Q09
LASTx Q10 SUM- Q11 SUMxy Q12 CLSUM Q13 RCL B Q14 SUM- Q15 LASTx Q16 SUM+ Q17 LASTx Q18 x^2
Q19 RCL A Q20 RCLx C Q21 - Q22 SUMx^2 Q23 + Q24 + Q25 x<0? Q26 GTO D Q27 SQRT Q28 RCL B Q29
ABS Q30 ENTER Q31 ÷ Q32 × Q33 STO E Q34 RCL+ B Q35 STO D Q36 RCL÷ A Q37 RCL D Q38 xNE0? x is
not equal to zero Q39 GTO C Q40 CLx Q41 STOP C01 LBL C C02 RCL C C03 x<>y C04 ÷ C05 STOP D01 LBL D
D02 SF 0 D03 +/- D04 SQRT D05 RCL÷ A D06 RCL B D07 RCL÷ A D08 STOP
```

A quadratic solver for the HP-32S which does not use the so-called "tricky properties" can still solve cases 4, 5 and 6 from the HP-15C Advanced Functions Handbook because those cases involve discriminants which are the difference between 11 significant digits. The reader can verify that if he must by deleting steps Q22 through Q24 in the program above. The 12 digit quadratic solver stumbles just a bit with case 8 in which the discriminant is the difference between the values $12,345,765,432.25 - 12,193,329,370 = 152,436,062.25$ and 0.05 of the difference is lost due to the 12 digit limitation. That loss of 0.05 in the discriminant changes one root from the correct value of 123,458 to 123,457.999998 and the other root from the correct value of 98,765 to 98,765.0000016. The complete program above solves case 8 correctly. I also devised two additional cases which require the full fifteen significant digits. Those two cases will not be solved correctly by a TI-59 single precision program or by a thirteen digit HP-41 program which incorporates the "tricky properties". Details for each of the test cases are included in the Appendix below.

APPENDIX - THE TEST CASE DETAILS

The quadratic equation used is $ax^2 - 2bx + c = 0$ for which the discriminant is $d = b^2 - ac$. Details are presented for cases 4 through 8 from the HP-15C Advanced Functions Handbook and for two additional cases which require a 15 digit capability.

Case 4:

$a = 654,323$ $b = 654,322$ $c = 654,321$

$$\begin{aligned}d &= (654,322)^2 - (654,323 \times 654,21) \\&= 428,137,279,684 - 428,137,79,683 = 1 \\R1 &= 1 \quad R2 = 0.999996943...\end{aligned}$$

Case 5:

$$\begin{aligned}a &= 11,713 \quad b = 735,246 \quad c = 46,152,709 \\d &= (735,246)^2 - (11,713 \times 46,152,709) \\&= 540,586,680,516 - 540,586,680,517 = -1 \\Re &= 62.77179203... ; Im = 8.5375E-05\end{aligned}$$

Case 6:

$$\begin{aligned}a &= 80,841 \quad b = 987,644 \quad c = 12,066,163 \\d &= (987,644)^2 - (80,841 \times 12,066,163) \\&= 975,440,670,736 - 975,440,683,083 = -12,347 \\Re &= 12.21711755... ; Im = 0.001374514...\end{aligned}$$

Case 7:

This case, which involves calculating the discriminant as the difference between two 20 digit values can be solved with the double precision program on pages 208-210 of the HP-15C Advanced Functions Handbook or by an equivalent program for the HP-41 which appears in my paper on the 13 digit capability of the HP-15C and HP-41.

$$\begin{aligned}a &= 4,877,361,379 \quad b = 4,877,262,613 \quad c = 4,877,163,849 \\d &= (4,877,262,613)^2 - (4,877,361,379 \times 4,877,163,849) \\&= 23,787,690,596,167,587,769 - 23,787,690,596,167,587,771 = -2 \\Re &= 0.999979750 ; Im = 2.8995463E-10\end{aligned}$$

Case 8:

$$\begin{aligned}a &= 1 \quad b = 111,111.5 \quad c = 12,193,329,370 \\d &= (111,111.5)^2 - (1 \times 12,193,329,370) \\&= 12,345,765,432.25 - 12,193,329,370 = 152,436,062.25 \\R1 &= 123,458 ; R2 = 98765\end{aligned}$$

The following additional test case is similar to case 4 from the HP-15C Advanced Functions Handbook. It yields a two digit difference between two fifteen digit numbers in the discriminant calculation. The positive value of the difference makes the roots real and unequal. Calculators such as (1) the HP-41 which typically retain only ten digits, (2) the HP-32S which typically retain only twelve digits, (3) the HP-41 programmed to use the “tricky properties”, and the TI-59 which typically retain only thirteen digits will all calculate the discriminant as zero and the two real roots will be erroneously found to be real and equal.

$a = 11,111,119$ $b = 11,111,111$ $c = 11,111,103$

$d = (11,111,111)^2 - (11,111,119 \times 11,111,103) =$

$= 123,456,787,654,321 - 123,456,787,257 = 64$

$R1 = 1$; $R2 = 0.999999280001$

This second additional test case is similar to case 5 from the HP-15C Advanced Functions Handbook. It yields a two digit difference between two 15 digit numbers in the discriminant calculation. The negative value makes the roots complex with $Re = b/a$ and $Im = \text{square root of } d \text{ divided by } a$ but if fewer than fourteen digits are retained due to calculator limitations then the discriminant will be calculated as zero and an incorrect answer of two equal real roots will result.

$a = 8,441,600$ $b = 11,111,111$ $c = 14,624,809$

$d = (11,111,111)^2 - (8,441,600 \times 14,624,909)$

$= 123,456,787,654,321 - 123,456,77,654,400 = - 79$

$Re = 1.31623282316$ $Im = \pm 7.19999489E-7$

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 67 Internals

Posted by [Tony Duell](#) on 2 Oct 2003, 9:42 p.m.

This machine may look like a 'classic series', but it uses an ACT (Arithmetic, Control, Timing) processor chip, and is electrically a Woodstock.

There are 3 PCBs in the case. The uppermost one (with the machine in the normal operating position) carries the keyboard, display, display drivers and card reader switch contacts. This is connected by 28 pins to the logic PCB that contains the processor, memory, card reader controller and power converter. Finally, this is connected by a 10 pin connector to the card reader PCB (very similar to that in the HP65).

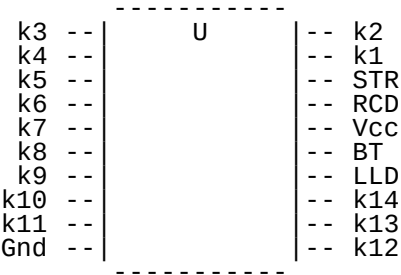
Connections from Keyboard to Logic PCB :

```
LHS of machine
0-- Vbsw
0-- Vbsw
0-- PGMSw/
0-- Phi_1
0-- Sync
0-- ISA
0-- KC1
0-- KC2
0-- KC3
0-- KC4
0-- KC5
0-- HDS/
0-- MTRS/
0-- RCD
0-- WPS
0-- STR
0-- SF
0-- SA
0-- SB
0-- SC
0-- SD
0-- SG
0-- SE
0-- SH
0-- Idisp
0-- Ibias
0-- Gnd
0-- Gnd
```

Vbsw : Switched Battery voltage, Gnd : Ground
PGMSw/ : Program/Run switch (low = Program mode)
Phi_1, Sync, ISA : Logic PCB testpoints (not connected on keyboard)
KCN : Keyboard column lines
HDS/ : Card at reader head
MTRS/ : Card reader motor switch
WPS : Card reader write protect (corner clipped) switch
RCD : Reset cathode driver, STR : Display digit strobe
Sx : Display segment drive lines
Idisp : Display current control, Ibias : control resistor

The keyboard PCB contains 2 ICs :

1820-1749 Cathode driver (also scans keyboard rows):
Essentially a shift register to select the kn lines in turn.



kn = Display cathode lines
Vcc = Switched battery voltage
RCD = reset cathode driver
STB = digit strobe
BT = Low battery control input
LLD = Low battery LED drive output

1858-0050 = CA3082 NPN transistor array
Yes, a standard IC in an HP handheld. It has 7 NPN transistors with a common collector (to the IDisp line). These drive the following segments:
Section Segment
a b
b e
c c
d f
e a
f h (decimal point)
g d

The g segment is driven by one of the discrete NPN transistors on the keyboard PCB, the other 3 transistors form a circuit to map both minus signs (g segments of figts 3 and 15) into the same digit. Note that the display is 'both ways scanned' -- a digit is selected, then the segments for that digit position are selected in turn.

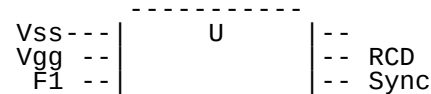
The keyboard is scanned by the cathode driver, 8 of the outputs select rows of the keyboard

Keyboard row	Cathode line
1	8
2	6
3	13
4	5
5	11
6	3
7	9
8	7

Logic PCB

Two of the discrete transistors here form the standard HP power converter circuit giving Vcc (+6.2V) and Vgg (-12V) from the battery. The other 2 form a current mirror circuit to set the display current.

1820-2530 : ACT processor



F2	--		--	PW0/
KC1	--		--	POR
KC2	--		--	Phi_1
KC3	--		--	Phi_2
KC4	--		--	
KC5	--		--	LC1
ISA	--		--	LC2
Data	--		--	Gnd

Vss : +6.2V, Vgg : =12V
 F1, F2 : Flag inputs (F1 N/C, F2 from card reader controller FLG line)
 KCh : Keyboard columns
 ISA : Instruction and Address signal to/from memory
 Data : Data line for RAM devices
 Sync : Bus sync line (Low during instruction part of the cycle, most of the time)
 Phi_n : 2 phase system clock (outputs on CPU)
 LCn : Connections for parallel LC clock tank circuit
 PW0/ : Power off
 POR : Power On Reset (conencted to Vss via 2.2uF capactior)

1818-0268 : ROM 0 and display decoder

SA	--		U		--	Sync
SF	--				--	Vgg
SB	--				--	Phi_2
SC	--				--	Phi_1
I	--				--	
SD	--				--	ISA
SH	--				--	Gnd
SE	--				--	STR
SG	--				--	Vss

Vss, Vgg : Power supply
 ISA : Instruction and Address
 Sync : System bus sync input
 Phi_n : 2 phase system clock
 STR : Display digit strobe
 Sx : Segment drive outputs (to display anode driver)
 I : Display current control input (connected to Vxx via resistor)

1818-0550, 1881-0551, 1818-0231, 1818-0232
 ROM/RAM ICs

Vss	--		U		--	Sync
Phi_2	--				--	ISA
Vgg	--				--	Gnd
Phi_1	--				--	Data

Vss, Vgg : Power supply
 ISA : Instruction and Address
 Sync : System bus sync input
 Phi_n : 2 phase system clock

Each chip contains ROM and 16 7 byte registers, so a total of 64 registers in the machine. 32 of them are used for 32*7 = 224 bytes (steps) of user program memory). The other 32 registers form the 26 user register (P0-P9, S0-S9, A-E and i), the 5 stack registers and one register containing the user flags, angle mode, decimal setting, etc.

1820-1751 Card Reader Controller

Motor/	--		U		--	Gnd
Vss	--				--	Flg/
WPS	--				--	
MTRS/	--				--	
HDS/	--				--	
PW0/	--				--	SF1
RB	--				--	
RA	--				--	



Vss, Vgg : Power supply
 ISA : Instruction and Address
 Sync : System bus sync input
 Phi_n : 2 phase system clock
 PW0/ : Power off
 HDS/ : Card at reader head
 MTRS/ : Card reader motor swtich
 WPS : Card reader write protect (corner clipped) switch
 SF1/ : Flag input (from Program switch)
 FLG/ : Flag output (to ACT F2)
 RA, RB : Card read data
 WA, WB : Card write data
 WE : Card write enable
 Motor/ : Motor on

Card reader to logic PCB connector :

Top of machine :

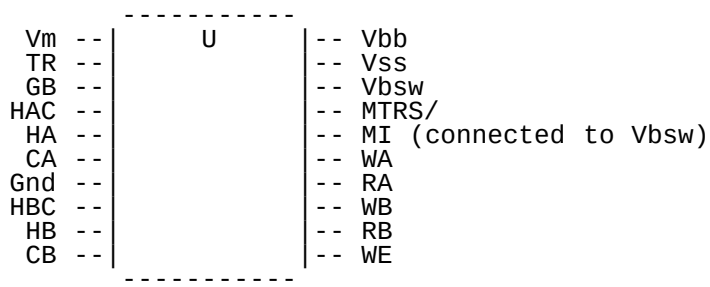
- o-- Vss
- o-- RA
- o-- RB
- o-- WE
- o--
- o-- WB
- o-- WA
- o-- VBSw
- o-- Motor/
- o-- Gnd

Vss, VBSw, Gnd : Power supply
 RA, RB : Card read data
 WA, WB : Card write data
 WE : Card write enable
 Motor/ : Motor on

Card reader

The 20 pin IC on the card reader PCB is the sense amplifier, write driver and motor controller. The pinout is :

1826-0322



Vm = Motor voltage output
 TR = Motor speed control resistor (other end to Vm)
 GB = Motor ground (to -ve side of battery pack and motor)
 Vbb = Unswitched battery voltage
 Vbsw = Switched battery voltage

Vss = +6.25V power input
MTRS/ = Motor run input
WA, WB = Write data inputs
RA, RB = Read data outputs
WE = Write Enable input
HA, HB = one side of head winding (for each track)
CA, CB = +ve end of 22uF cap (other side to HAC, HBC)
HAC, HBC = -ve side of 22uF cap, other end of head winding.

Head wiring colours :

Blue = HA }
Orange = HAC } Track A winding

Red = HB }
Yellow = HBC } Track B winding

Black = ground

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

HP Charger Data

Posted by [Tony Duell \(UK\)](#) on 18 Sept 2003, 9:25 p.m.

HP Chargers : 220 and 240V versions run from UK 240V mains. 110V versions run from 110V transformer.

Woodstock :
HP82025A (UK, 240V) : Open Circuit Voltage = 10.42V, R(int) = 14.51 Ohms
HP82026A (dual, 115V) : Open Circuit Voltage = 9.95V, R(int) = 12.49 Ohms
HP82026A (dual, 230V) : Open Circuit Voltage = 10.14V, R(int) = 11.93 Ohms

Spice
HP82087B (USA, 110V) : Open Circuit Voltage = 10.14V, R(int) = 13.70 Ohms
HP82089A (UK, 240V) : Open Circuit Voltage = 10.52V, R(int) = 13.93 Ohms
HP82089B (UK, 240V) : Open Circuit Voltage = 10.44V, R(int) = 14.08 Ohms
HP82090B (Europe, 220V) : Open Circuit Voltage = 10.26V, R(int) = 14.34 Ohms

Standard
HP82059B (USA, 110V) : Open Circuit Voltage = 12.17V, R(int) = 10.85 Ohms
HP82066B (Europe, 220V) : Open Circuit Voltage = 13.38V, R(int) = 11.66 Ohms
HP82067B (UK, 240V) : Open Circuit Voltage = 12.68V, R(int) = 11.36 Ohms

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

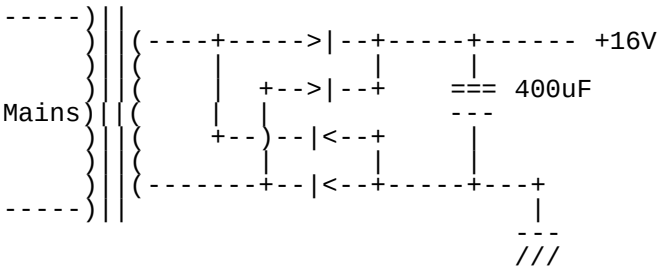
[Previous | Next]

Classic Charger Schematic

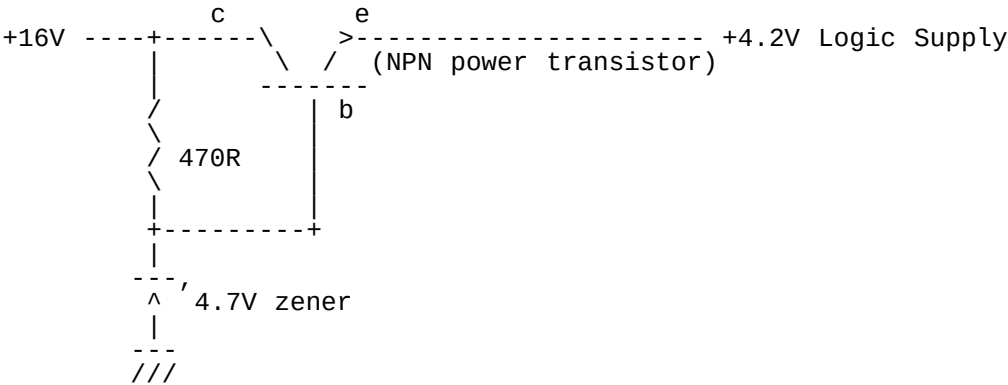
Posted by [Tony Duell \(UK\)](#) on 18 Sept 2003, 9:23 p.m.

HP82011 (etc) 'classic series' charger

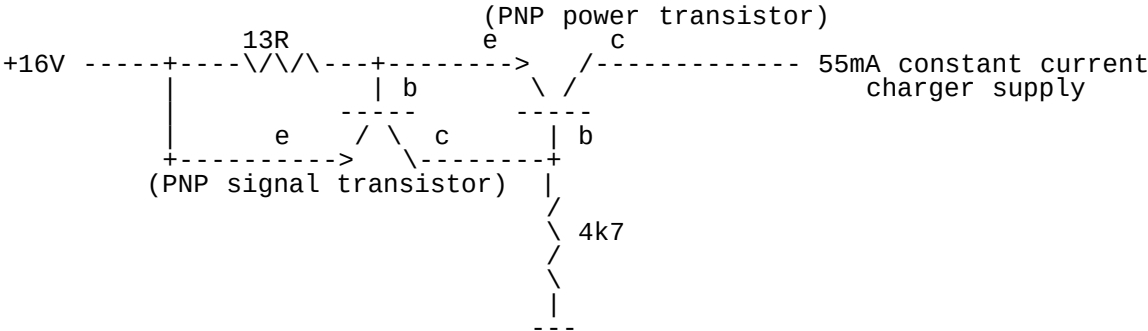
(Unregulated PSU)



(Constant voltage regulator)



(Constant current charger)



///

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 65 Internals

Posted by [Tony Duell \(UK\)](#) on 18 Sept 2003, 9:22 p.m.

There are 3 PCBs in the case. The lowest one contains the keyboard switches, displays, and display drivers. It connects by a 28 pin connector to the logic PCB, which contains the PSU, CPU hybrid and ROMs. The logic PCB is then connected to the card reader PCB by a 10 pin connector. The card reader head and motor are hardwired to the card reader PCB.

The 28 pin connector between the keyboard/display and the logic PCB carries similar signals to the 25 pin connector in the HP35, together with 3 switch signals from the card reader contacts and the program switch signal. The pinout is :

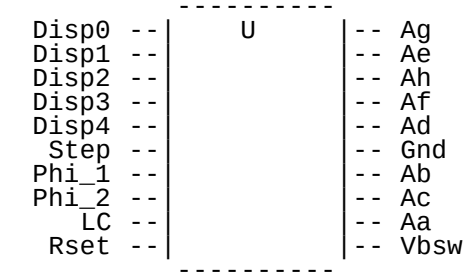
1 (LHS of machine)

```
0-- Disp 0
0-- Disp 1
0-- Disp 2
0-- KC1
0-- Disp 3
0-- Disp 4
0-- HDS/
0-- WPS
0-- MTRS/
0-- Phi_1
0-- KC2
0-- Vbsw
0-- Phi_2
0-- Gnd
0-- Vcc
0-- PGMSw/
0-- KC3
0-- KR1
0-- KR2
0-- KR3
0-- KC5
0-- KR4
0-- KC4
0-- KR5
0-- KR6
0-- KR7
0-- KR8
0-- RCD
```

KC = keybaord column line, KR = keyboard row line
 Disp = display data bus from CPU hybrid to anode driver
 RCD = Reset cathode driver (start display scan)
 Phi_n = 2 phase clock from anode driver to CPU
 Vcc = 8.2V power line to cathode driver
 Vbsw = Switched battery voltage
 PGMSw/ = program mode switch (ground = w/pgm mode)
 HDS/ = card reader head swtich (ground = card over head)
 MTRS/ = card reader motor switch (ground = card inserted)
 WPS = card reader write protect switch (ground when HDS/ goes to ground = corner not clipped)

There are 2 custom chips on the keyboard/display PCB :

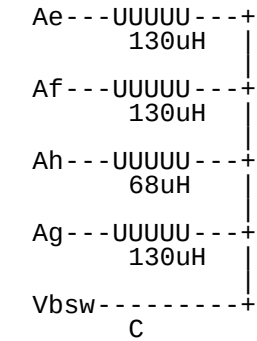
1820-1029 (anode driver and system clock)



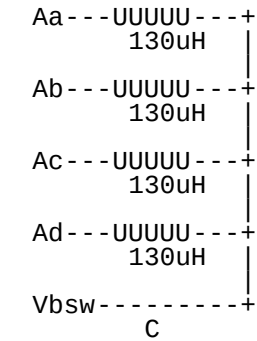
Disp = Display data bus (as above).
A = display anode drive lines [1], segments named conventionally (h = decimal point)
Phi_n = clock output to logic board
Step = step output to cathode driver (display scanning)
LC = oscillator components (LC series circuit from here to ground)
Rset = resistor from here to ground, appears to be part of low battery circuit

[1] There's an inductor of 130uH (or 68uH for the point) from each anode line to Vbsw. Display runs from back-emf from these inductors. The HP65 uses 4-inductor hybrid modules rather than the discrete inductors used in earlier machines :

1810-0146



1810-0147



1820-1226 cathode driver (essentially a 15 stage shift register for display scanning).

k9 --	U	-- k8
k10 --		-- k7
k11 --		-- k9
Vbsw --		-- k5
k12 --		-- k4
k13 --		-- Vbsw
k14 --		-- k3
RCD --		-- k2
Step --		-- k1
Vcc --		-- k15

k = display cathode outputs. Step = display scan clock from anode driver
RCD = Reset Cathode Driver = start scan (from A&R chip)
Note lack of ground pin -- runs between Vcc (8.2V) and Vbsw (4V).

Logic PCB

The discrete components on this board are almost all part of the PSU, which is very similar to that in earlier machines. It provides Vss (+6.25V) and Vgg (-12V) to the logic, and Vcc (+8.2V) to the display driver.

The CPU in this machines is a 44 pin hybrid circuit containing 6 chips. These are believed to be :

Arithmetic and Registers

Control and Timing
2 * RAM (data and user program store?)
Clock driver and init
Card Reader Controller.

The pinout of the hybrid (60218-00065) is :

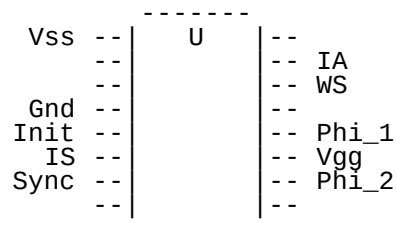
WE --	U	-- RB
WPS --		-- RA
T/P --		--
HDS/ --		-- Phi_2 In
MTRS/ --		-- Phi_1 In
WB --		-- Phi_2 Out
WA --		-- Phi_1 Out
IA --		-- Init
Gnd --		-- Init C
Vss --		-- Data
Disp2 --		-- RCD
Disp1 --		-- IS
Disp0 --		-- Sync
Disp3 --		-- PGMSw/
Disp4 --		-- WS
Vgg --		-- KC1
KR2 --		-- KC2
KR3 --		-- KC3
KR4 --		-- KC4
KR5 --		-- KC5
KR6 --		-- KR7
KR1 --		-- KR8

KR = Keyboard row lines, KC = keyboard column lines
Disp = display data (to anode driver)
RCD = Reset Cathode Driver
Phi_n In = 2 phase clock input from anode driver
Phi_n Out = 2 phase clock output to ROMs
Init C = Reset capacitor input (2.2uF from this pin to ground)
Init = Reset output to ROMs
Sync = machine cycle synchronisation
IA = Instruction Address to ROMs
IS = Instruction from ROMs
WS = Word select
Data = serial BCD data from X register (testpoint)
Vss = +6.25V, Vgg=-12V power lines

PGMSw/ = program switch input
HDS/, MTRS/, WPS = card reader switch inputs (see above)
RA, RB = Read data inputs from the 2 card reader tracks
WA, WB = Write data outputs to card reader tracks
WE = Card reader write enable
T/P = testpoint (connected to card reader connector)

The other 3 chips on the logic PCB are ROMs :

1818-0073, 1818-0074, 1818-0075



The signals are as described above.

There are 10 testpoints on the logic PCB :

(Top of machine)

0-- Sync
0-- IS
0-- IA
0-- Vgg
0-- Phi_2 Out
0-- Phi_1 Out
0-- Init
0-- Data
0-- WS
0-- RCD

There is also a 10 pin connector to the card reader :

(Top of machine)

0-- Vss
0-- RA
0-- RB
0-- WE
0-- T/P (from CPU pin 3)
0-- WB
0-- WA
0-- Vbsw
0-- MTRS/
0-- Gnd

RA, RB = Read data from card reader to CPU (2 tracks)
WA, WB = Write data from CPU to card reader (2 tracks)
WE = Write Enable
Vbsw = switched battery voltage
Vss = +6.25V
MTRS/ = motor switch (to start card reader motor)

Card reader

The 20 pin IC on the card reader PCB is the sense amplifier, write driver and motor controller. The pinout is :

1826-0158

Vm	--	U	--	Vbb
TR	--		--	Vss
GB	--		--	Vbsw
HAC	--		--	MTRS/
HA	--		--	MI (connected to Vbsw)
CA	--		--	WA
Gnd	--		--	RA
HBC	--		--	WB
HB	--		--	RB
CB	--		--	WE

Vm = Motor voltage output
TR = Motor speed control resistor (other end to Vm)
GB = Motor ground (to -ve side of battery pack and motor)
Vbb = Unswitched battery voltage
Vbsw = Switched battery voltage
Vss = +6.25V power input
MTRS/ = Motor run input
WA, WB = Write data inputs
RA, RB = Read data outputs
WE = Write Enable input
HA, HB = one side of head winding (for each track)
CA, CB = +ve end of 22uF cap (other side to HAC, HBC)
HAC, HBC = -ve side of 22uF cap, other end of head winding.

Head wiring colours :

Blue = HA)
Orange = HAC) Track A winding

Red = HB)
Yellow = HBC) Track B winding

Black = ground

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

HP 55 Internals

Posted by [Tony Duell \(UK\)](#) on 18 Sept 2003, 9:19 p.m.

Mechanical construction (and architecture) very similar to HP45 (see previous notes). 2 PCBs, keyboard and display on upper PCB, CPU and ROMs on lower PCB. Connections between PCBs by 28 pins

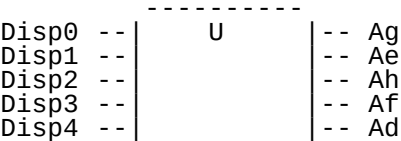
```
0-- Disp 0
0-- Disp 1
0--
0-- KC1
0-- Disp 2
0-- Disp 3
0-- Disp 4
0-- Phi_1
0-- Phi_2
0-- KC2
0-- Vbsw
0-- Vcc
0-- Gnd
0-- Timer/
0-- PGM/
0-- RCD
0-- KC3
0-- KR1
0-- KR2
0-- KR3
0-- KC5
0-- KR4
0-- KC4
0-- KR5
0-- KR6
0-- KR7
0--
0-- KR8
```

KC = Keyboard Column line. KR = Keyboard Row line
Disp = display data bus from A&R chip to anode driver
RCD = Reset Cathode Driver - start display scan
Timer/ = connected to ground in 'Timer' mode
PGM/ = connected to ground in PGM mode
Phi_n = 2 phase system clock
Vcc = 8.2V power line from PSU on logic board to cathode driver. Vbsw = switched battery voltage.

Keyboard/display PCB

There are 2 custom chips on the keyboard/display PCB.

1820-1029 (anode driver and system clock)

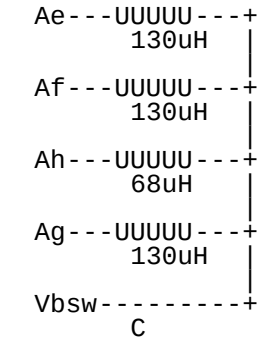




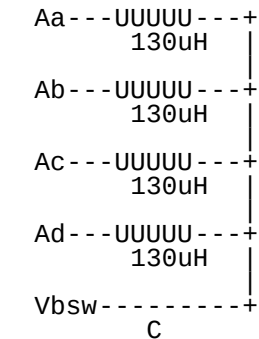
Disp = Display data bus (as above).
A = display anode drive lines [1], segments named conventionally (h = decimal point)
Phi_n = clock output to logic board
Step = step output to cathode driver (display scanning)
LC = Xtal from here to ground (~ 784kHz)
Rset = resistor from here to ground, appears to be part of low battery circuit

[1] There's an inductor of 130uH (or 68uH for the point) from each anode line to Vbsw. Display runs from back-emf from these inductors. The HP55 uses 4-inductor hybrid modules rather than the discrete inductors used in earlier machines :

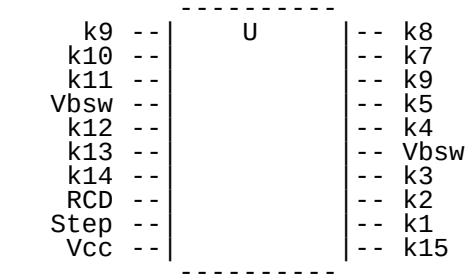
1810-0146



1810-0147



1820-1226 cathode driver (essentially a 15 stage shift register for display scanning).

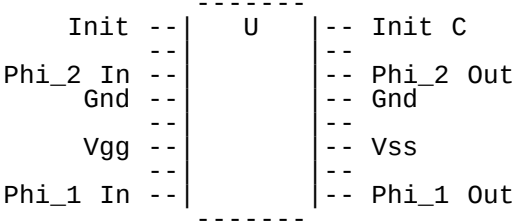


k = display cathode outputs. Step = display scan clock from anode driver
RCD = Reset Cathode Driver = start scan (from A&R chip)
Note lack of ground pin -- runs between Vcc (8.2V) and Vbsw (4V).

Logic PCB

The 2 transistors and the transformer form a power supply giving the same 3 voltages (for the same purposes) as the HP35. There is no discrete component reset circuit, though (see below for what replaced it).

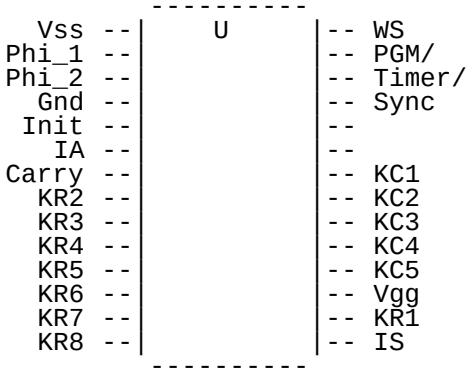
```
1820-1128 Clock driver and init
```



This replaces the 8 pin clock driver of the HP35 and most of the discrete components in the reset circuit. The signals are :

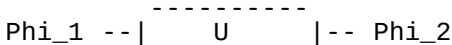
```
Phi_n In : 2 phaseclock inputs from anode driver (at 'bipolar' levels)
Phi_n Out (called Phi_n on other logic board chips) : 2 phase clock
outputs to rest of CPU
Init : Reset output to C&T and ROMs
Init C : 2.2uF capacitor from here to ground -- reset timing capacitor.
This is the only component in the HP35 reset circuit that couldn't be put
on the chip.
```

Control & Timing (C&T) -- the control part of the CPU (and keyboard scanner)
1818-0078



KC = Keyboard Column, KR = Keyboard Row (to keyboard matrix)
 Carry = Carry flag input from A&R chip
 Init = power-on reset
 IA = Instruction Address = address to ROMs
 IS = InStruction = ROM Data input
 Sync = machine cycle sync signal
 Phi = 2 phase clock
 WS = word Select from ROMs (used to update part of a 56 bit word)
 PGM/ = pulled low in PGM mode (and pulled low via diode in timer mode)
 Timer/ = pulled low in timer mode

Arithmetic & Registers (A&R) -- CPU data path
1820-1169



Vss	--				--	Gnd
IS	--				--	Vgg
Sync	--				--	Disp4
WS	--				--	Disp3
Carry	--				--	Disp0
Data	--				--	Disp1
RCD	--				--	Disp2

Disp = Display data bus. RCD = Reset Cathode Driver.
 Phi = 2 phase clock, IS = InStruction (from ROMs)
 Sync = machine cycle sync signal
 Carry = Carry flag output
 WS = Word Select
 Data = bit-serial contents of X (display) register.

1818-0118, 1818-00119, 1818-0120 : ROMs

Similar to the ROMs in the HP35, but presumably higher capacity. Now packaged in 16 pin DIL (ceramic) packages. The signals are the same as the HP35 ROM interface, and the pinout is :

Vss	--		U		--	
	--				--	IA
	--				--	WS
Gnd	--				--	
Init	--				--	Phi_1
IS	--				--	Vgg
Sync	--				--	Phi_2
	--				--	

See above for signal descriptions

1820-1393 RAM (3 used, address selected by SelA, SelB pins)

This connects to the data pin on the A&R chip and transfers data to/from the arithmetic registers.

The pinout is :

Vss	--		U		--	IS
Gnd	--				--	
Gnd	--				--	Data
Gnd	--				--	Phi_2
SelB	--				--	Phi_1
SelA	--				--	Vgg
Gnd	--				--	
Gnd	--				--	RCD

Data : Data to/from A&R chip
 Phi_n : 2 phase clock inputs
 IS : ROM output (InStruction) line, used to address RAM.
 RCD : Reset Cathode Driver signal from A&R, used to define start of a word.
 SelA, SelB : Select address range for this RAM (both low on 1st RAM, Sel A connected to Vss on one of the other 2 RAMs, SelB connected to Vss on last RAM)

Testpoints ;
 There are 12 testpoints on the edge of the CPU board, in the same position and carrying the same signals as on the HP35. For reference, the signals are :

0-- Vss
 0-- Gnd
 0-- Phi_1
 0-- Phi_2
 0-- Vgg
 0-- IS

o-- IA
o-- Sync
o-- WS
o-- Init
o-- Carry
o-- Data

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 45 Internals

Posted by [Tony Duell \(UK\)](#) on 18 Sept 2003, 9:18 p.m.

Mechanical construction (and architecture) very similar to HP35 (see previous notes). 2 PCBs, keyboard and display on upper PCB, CPU and ROMs on lower PCB. Connections between PCBs by 25 pins (same signals as HP35).

Keyboard/display PCB

This has an 00035- part number, and seems to be electically identical to the board in an HP35. The 2 chips have different numbers :

Anode driver : 1820-1029
Cathode driver : 1820-1226

but seem electically very similar to the chips in an HP35. Same pinouts (so see previous notes again).

Logic PCB

The 2 transistors and the transformer form a power supply giving the same 3 voltages (for the same purposes) as the HP35. There is no discrete component reset circuit, though (see below for what replaced it).

2 of the chips :

Control and Timing : 1818-0078
Arithmetic and Registers : 1820-1169

have the same pinout as the corresponding chips on the HP35 logic PCB. Presumably they have different instruction sets, though.

This leaves 4 more chips :

1820-1128 Clock driver and init

Init	--	U	--	Init C
	--		--	
Phi_2 In	--		--	Phi_2 Out
Gnd	--		--	Gnd
	--		--	
Vgg	--		--	Vss
	--		--	
Phi_1 In	--		--	Phi_1 Out

This replaces the 8 pin clock driver of the HP35 and most of the discrete

components in the reset circuit. The signals are :

Phi_n In : 2 phase clock inputs from anode driver (at 'bipolar' levels)
 Phi_n Out (called Phi_n on other logic board chips) : 2 phase clock
 outputs to rest of CPU
 Init : Reset output to C&T and ROMs
 Init C : 2.2uF capacitor from here to ground -- reset timing capacitor.
 This is the only component in the HP35 reset circuit that couldn't be put
 on the chip.

1818-0057, 1818-0082 : ROMs

Similar to the ROMs in the HP35, but presumably higher capacity. Now
 packaged in 16 pin DIL (ceramic) packages. The signals are the same as
 the HP35 ROM interface, and the pinout is :

```

      Vss --|      U      |--
      --|      |      |-- IA
      --|      |      |-- WS
      Gnd --|      |      |--
      Init --|      |      |-- Phi_1
      IS --|      |      |-- Vgg
      Sync --|      |      |-- Phi_2
      --|      |      |--
    
```

See the HP35 notes for signal descriptions

1820-0993 RAM

This connects to the (unused in the HP35) data pin on the A&R chip and
 transfers data to/from the arithmetic registers.

The pinout is :

```

      Vss --|      U      |-- IS
      Gnd --|      |      |--
      Gnd --|      |      |-- Data
      Gnd --|      |      |-- Phi_2
      Gnd --|      |      |-- Phi_1
      Gnd --|      |      |-- Vgg
      Gnd --|      |      |--
      Gnd --|      |      |-- RCD
      Gnd --|      |      |--
    
```

Data : Data to/from A&R chip
 Phi_n : 2 phase clock inputs
 IS : ROM output (InStruction) line, used to address RAM.
 RCD : Reset Cathode Driver signal from A&R, used to define start of a word.

Testpoints ;

There are 12 testpoints on the edge of the CPU board, in the same
 position and carrying the same signals as on the HP35. For reference, the
 signals are :

```

0-- Vss
0-- Gnd
0-- Phi_1
0-- Phi_2
0-- Vgg
0-- IS
0-- IA
0-- Sync
0-- WS
0-- Init
0-- Carry
0-- Data
    
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

HP 35 internals

Posted by [Tony Duell \(UK\)](#) on 18 Sept 2003, 9:15 p.m.

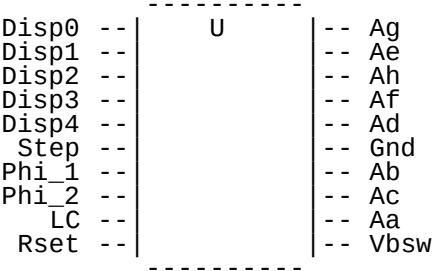
There are 2 PCBs, connected by 25 pin contacts. The upper PCB contains the display, keyboard, display drivers, etc. The lower PCB contains the processor and firmware ROMs. The connections between the PCBs are :

- 0-- Disp 0
- 0-- Disp 1
- 0-- KC1
- 0-- Disp 2
- 0-- Disp 3
- 0-- Disp 4
- 0-- Phi_1
- 0-- Phi_2
- 0-- KC2
- 0-- Vbsw
- 0-- Vcc (+8.2V)
- 0-- Gnd
- 0-- Gnd
- 0-- RCD
- 0-- KC3
- 0-- KR1
- 0-- KR2
- 0-- KR3
- 0-- KC5
- 0-- KR4
- 0-- KC4
- 0-- KR5
- 0-- KR6
- 0-- KR7
- 0-- KR8

KC = Keyboard Column line. KR = Keyboard Row line
Disp = display data bus from A&R chip to anode driver
RCD = Reset Cathode Driver - start display scan
Phi_n = 2 phase system clock
Vcc = 8.2V power line from PSU on logic board to cathode driver. Vbsw = swiotched battery voltage.

There are 2 custom chips on the keyboard/display PCB.

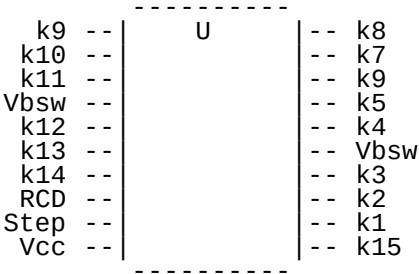
1820-0853 (anode driver and system clock)



```
Disp = Display data bus (as above).
A = display anode drive lines [1], segments named conventionally (h = decimal point)
Phi_n = clock output to logic board
Step = step output to cathode driver (display scanning)
LC = oscillator components (LC series circuit from here to ground)
Rset = resistor from here to ground, appears to be part of low battery circuit
```

[1] There's an inductor of 130uH (or 68uH for the point) from each anode line to V_{sw}. Display runs from back-emf from these inductors when the anode driver turns off. Neat!

1820-0854 cathode driver (essentially a 15 stage shift register for display scanning).



k = display cathode outputs. Step = display scan clock from anode driver
RCD = Reset Cathode Driver = start scan (from A&R chip)
Note lack of ground pin -- runs between Vcc (8.2V) and Vbsw (4V).

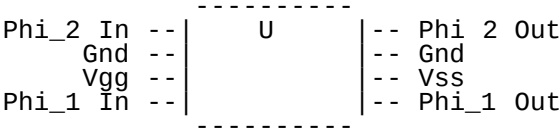
Logic PCB

Firstly the discrete components. The 2 transistors above the C&T chip (28 pin) are the PSU. They, together with the pulse transformer, provide 3 voltages to the rest of the machine : Vss (+6V), Vgg (-12V) and Vcc (+8.2V)

The other 2 transistors, and associated components, form a schmitt trigger for the power-on reset (init) signal. This is fed to the C&T chip and the ROMs.

The little 8 pin chip :

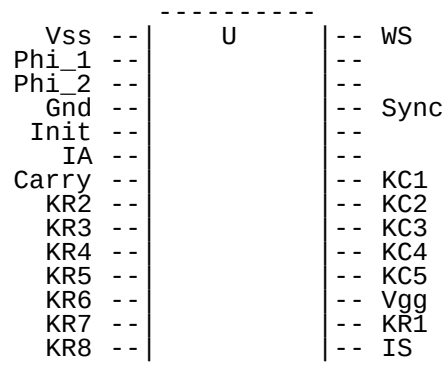
1820-0855 Clock driver



Buffers clock signals (at 6V levels) from anode driver to MOS levels for rest of CPU logic. From now on, Phi_n will mean the MOS signals!

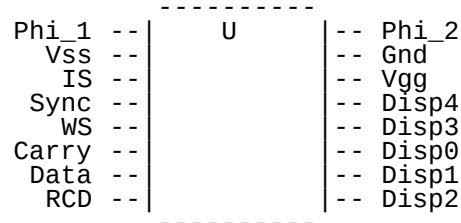
The processor is in 2 chips, basically control (C&T) and data path (A&R). It operates on 56 bit words (which hold a normalised real number in BCD).

Control & Timing (C&T) -- the control part of the CPU (and keyboard scanner)
1820-0849



KC = Keyboard Column, KR = Keyboard Row (to keyboard matrix)
 Carry = Carry flag input from A&R chip
 Init = power-on reset
 IA = Instruction Address = address to ROMs
 IS = InStruction = ROM Data input
 Sync = machine cycle sync signal
 Phi = 2 phase clock
 WS = Word Select from ROMs (used to update part of a 56 bit word)

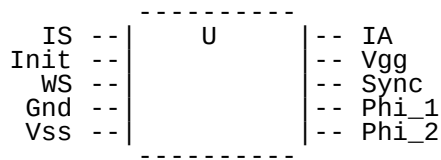
Arithmetic & Registers (A&R) -- CPU data path
 1820-0848



Disp = Display data bus. RCD = Reset Cathode Driver.
 Phi = 2 phase clock, IS = InStruction (from ROMs)
 Sync = machine cycle sync signal
 Carry = Carry flag output
 WS = Word Select
 Data = bit-serial contents of X (display) register. Only used as T/P in the HP35

ROMs : These are 256 word (10 bit word) devices, serial address and data, in 10 pin T099 cans (can't show in ASCII, so will show as 10 pin DIP).

1818=0024, 1818-0026, 1818-0028



Signals as above (C&T chip)

Testpoints : There are 12 testpoints at one edge of the logic PCB, near the A&R chip. Used for factory testing/diagnosis. Just about all signals are available either here or on the keyboard/display connector, so it's possible to find most faults from the signals here. The testpoints are :

```
o-- Vss
o-- Gnd
o-- Phi_1 } Output from clock driver
o-- Phi_2 }
o-- Vgg
o-- IS
o-- IA
o-- Sync
o-- WS
o-- Init
o-- Carry
o-- Data
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-97S Test Connector

Posted by [Erik Wahlin \(USA\)](#) on 6 Aug 2003, 9:37 p.m.

I had a few requests to map out my HP-97S test connector. It is a fairly simple thing to build if you can find the IC.

A possible vendor of the IC is [Emar International](#) who list them on their web site.

Also, Katie Wasserman noted that "The MM80C98 is the same at the MM70C98 which is still made by [National](#) Also the MM80C98 isn't that hard to find. [Mouser](#) has them for \$2.37."

Please see below or [my site](#) for the connector details. I have attempted as best as I can to double and triple check the connections. Please use at your own risk however. I assume no liability if you damage your HP-97S by building and using this connector. The HP-97S diagnostic program that is used with this connector is also shown in the [here](#).

97S Test Connector IC

16 15 14 13 12 11 10 9

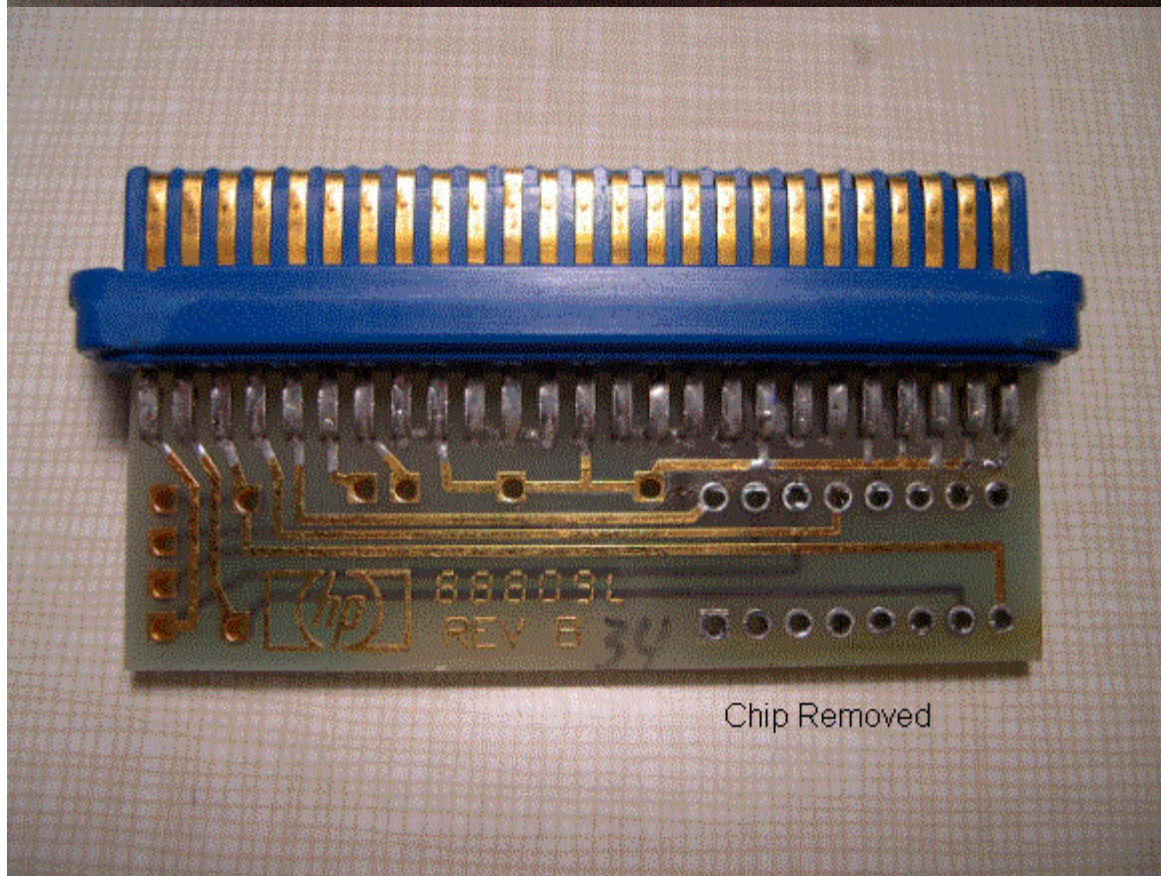
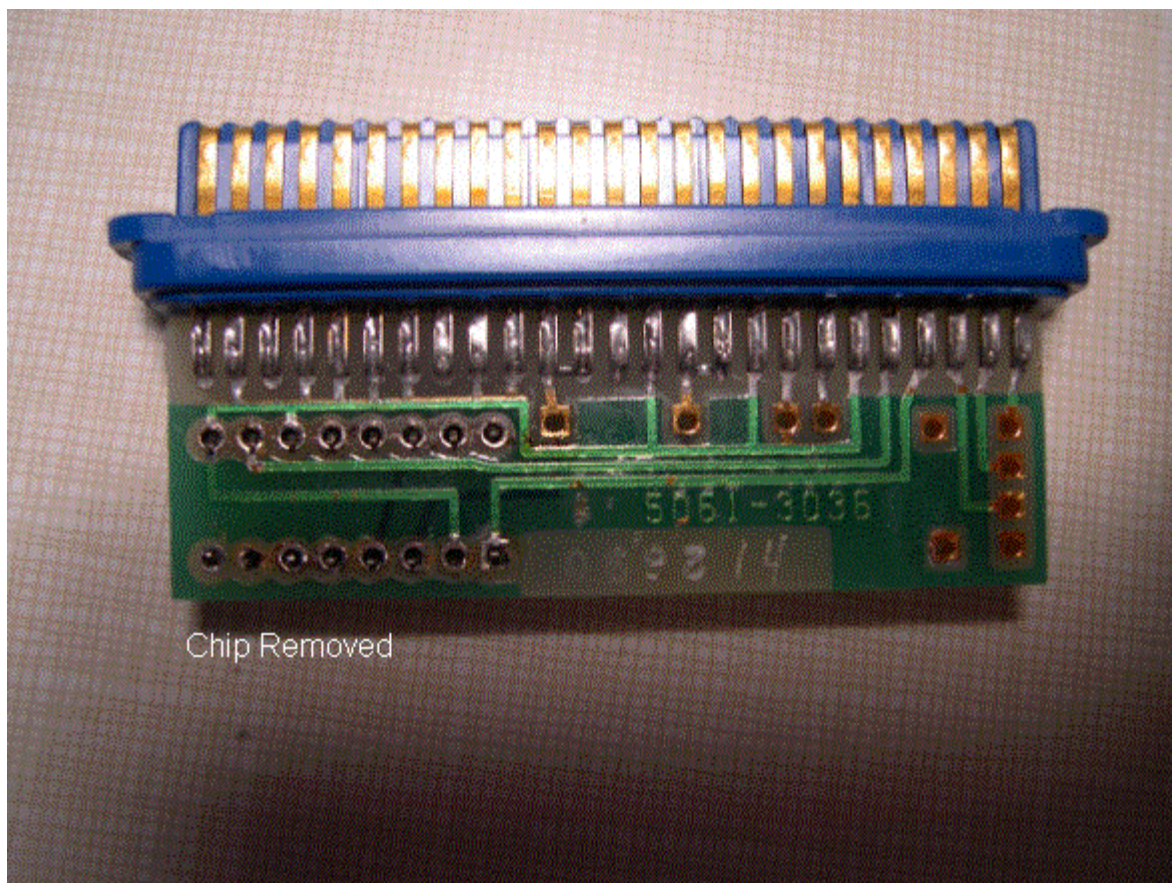


1 2 3 4 5 6 7 8

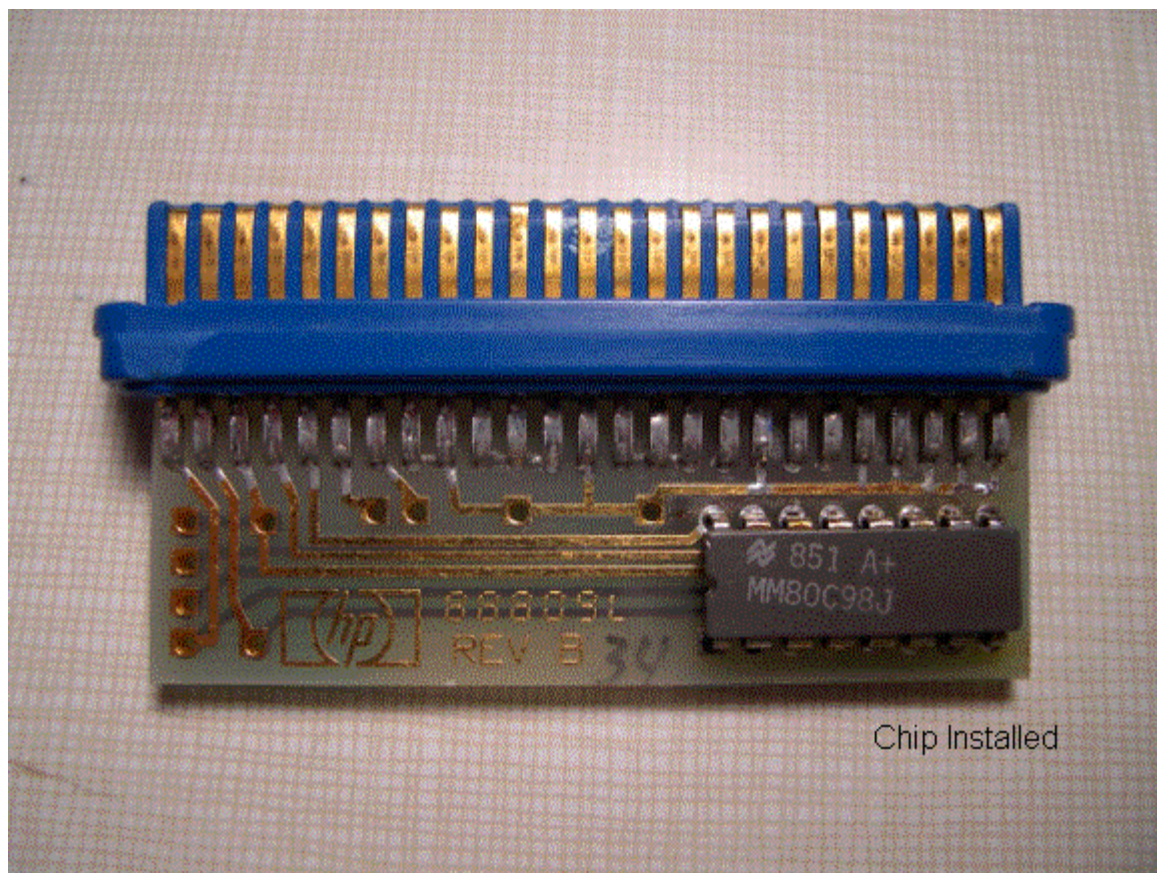
Amphenol

Pin #	Connections to 50 pin Amphenol (57-30500) connector
1	28, IC8
2	IC12
3	IC15
4	IC1
5	IC10
6	8, 9, 12, 16, 17, 19, 20, 21, 22, 24, 32, 35, 36, 37, 39, 40, 41, 42, 44, 45, IC2, IC9, IC11
7	10, 11, 13, 14, 15, 18, 23, 25, 31, 33, 34, 38, 43, 46, 47, 48, 49, 50, IC3
8	6
9	6
10	7
11	7
12	6
13	7
14	7
15	7
16	6
17	6
18	7
19	6
20	6
21	6
22	6
23	7
24	6
25	7
26	IC4, IC14
27	IC7
28	1
29	IC13
30	IC16
31	7
32	6
33	7
34	7
35	6
36	6
37	6
38	7
39	6
40	6
41	6
42	6
43	7
44	6
45	6
46	7
47	7
48	7
49	7
50	7
Chip Connections (Not connected to 50 pin connector)	
IC5, IC6	

Pictures with chip removed



Chip Installed



Edited: 6 Aug 2003, 10:03 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

HP-15C First Degree Runge-Kutta Order 4

Posted by [Patrick](#) on 4 Aug 2003, 2:56 p.m.

Description

Here is an HP-15C program that implements a fourth-order Runge-Kutta algorithm for a first degree ODE. It has the attributes of being somewhat miserly in its use of resources compared with routines published in HP Solutions handbooks. The program is only 45 steps long and requires the use of only 4 registers and 1 flag.

Usage

Label A

Input: h ENTER y ENTER x [A]

Output: X = new x, Y = new y, Z = delta X = h, T = delta Y

Label B

Assumes: Label A previously run.

Input: h [B]

Output: Same as for Label A

Label C

Assumes: Label A previously run.

Input: Nothing.

Output: Same as for Label A.

Label D (not provided)

This is the label under which you must code the function, $f(x,y)$, which defines the derivative: $y' = f(x,y)$.

Basically, Label A is used to initialize the algorithm and compute a single step. Label C is used to take additional steps with the *same* step size. Label B is used to take additional steps with a *new* step size (for use with adaptive step-size techniques).

Resources Used

R0 = h/2
R1 = x
R2 = y
R3 = Used

Flag 0 = Used, left clear

Program

```

001      LBL A    Main algorithm entry point specifying x, y, and h
002      STO 1    Initialize R1 = x
003      RDN
004      STO 2    Initialize R2 = y
005      RDN
006      LBL B    Entry point for adaptive step size techniques.
007      2
008      /
009      STO 0    Initialize R0 = h/2
010      LBL C    Entry point for RK iteration using current values
011      CLX
012      STO 3
013      CF 0
014      0
015      GSB 0
016      SF 0
017      1
018      GSB 0
019      1
020      GSB 0
021      CF 0
022      2
023      GSB 0
024      RCL 3
025      3
026      /
027      STO+2
028      2
029      RCL*0
030      STO+1
031      RCL 2
032      RCL 1
033      RTN
034      LBL 0    Internal label
035      *
036      RCL+2
037      LAST x
038      RCL*0
039      RCL+1
040      GSB D    Derivative at specified (x,y) point
041      RCL*0
042      STO+3
043      FS?0
044      STO+3
045      RTN
046      LBL D
Code derivative function here
      RTN

```


Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Service tips for new owners of HP-34C, other spice calc's

Posted by [Norm](#) on 5 June 2003, 9:02 p.m.

HELLO:

I have something to say. It's my best information to you, about servicing the HP-34C. Along with other people's suggestions, these ideas should give you enough knowledge to keep it running right.

OF course, we have the great division between these units, of solderless assembly (the chips are merely pressed against the wiring board) and a later subsequent soldered version which seems less common.

SO, the solderless units were a bad idea originally, and you may well have one that becomes unreliable.

For starters, know which version you have. YES, that means opening it up and taking a look-see. NO, I am not going to get into the frightful details of how to separate the lower-edge of the case halves. It is possible but its difficult and that's another story.

What I will earnestly plead... if you have a solderless unit, and it starts getting unreliable on you, STOP using it. I am convinced that a degradation takes place in these units, if the solderless joints are intermittent and you keep on using it. If its healthy, fine, enjoy it (if it aint broke dont fix it) once you see evidence of trouble then take it to the doctor.

As to how to bring a solderless unit back to fullest performance (if the connections go intermittent) I have some answers there, though maybe not all.

I tried repairing one by soldering it with a controlled temperature precision iron. I think it wasn't the best idea, so you can learn from my experiences:

Firstly, the pins themselves may have built up a blue oxide, esp. on the 3 small memory chips. This oxide not only may contribute to the bad connections, butit also prevents you from soldering the pin down. The pins don't want to accept solder.

So they weren't tinning. The only way to fix that was flip the chip on its backside, and rub it for awhile with the iron.

Secondly, even with the lowest controlled temperature, there was a tendency to melt the plastic underneath the connection.

Thirdly, you can never undo this type of work, so if a chip blows you can't replace it and the supply of parts rapidly dwindles.

I would favor the Luiz Vieira approach..... pull it apart and clean up the contact surfaces with a cleaner. Then use

some type of light-oil that would keep the surfaces more corrosion resistant (dont know what oil, probably a switch-lube oil.

The chips themselves, well, flip them upside down, and look for cleanliness of each pin. Probably a bit of scraping with a jewelers screwdriver is all that's needed. I think Luiz was using a bit of emery paper, which might be OK if followed by a squirt of aerosol degreaser to make the abrasives go away.

A low-power stereozoom microscope is just dynamite for this. You can see exactly what's going on with each pin. I have one..... although they are \$800 new, it can be such a fine tool that I say its totally worth the money. It's a Leica Zoom 2000, stereo, 7 - 30 power with built-in illuminators.

THE LAST THING is ESD. I dont think you stand a snowballs chance in Hell of being able to clean up all the pins on an ancient CMOS chip without frying it. The only way this will work is if you ground yourself out, like by stuffing aluminum foil into your socks, ground the jewelers screwdriver out, have everything grounded using alligator clips, and work on a piece of anti-static plastic or perhaps a piece of aluminum foil while you work each pin of a chip on its backside. Even that might be a bad recipe, often ESD is best handled with weakly conductive paths, not strong ones. Pick your strategy and run with it. I'd probably go with hard-metal connections for a job like this, especially to myself, and work atop aluminum foil.

Anyway, clean all these boundaries where the contacts occurs, a bit of oil on everything, put it back together and cross your fingers.

That's the Luiz approach, and its better than what I tried.

Do recommend lubrication of the switches, or they will grind flakey bits of metal where the contacts are sliding. And these are antiques that there's no turning back (unless this whole country gets more traditional in its thinking, from the boardroom on down). So we really need to cherish these items.

I think we can take a line from the antique radio crowd..... "you are dealing with a nostalgic item that goes beyond just your personal ownership. Later it will be owned by another person. Do any work so that it will last andould please the future owners also".

Still recommend "Tri Flow" for slide switches (the teflon version of WD-40, from the black aerosol can), shake well, but only one drop of the liquid onto a Q-tip and then rub it onto the slide switches while everything is apart.

HEY one OTHER VERY IMPORTANT IDEA about SPICE calculators (dont think people have bought into this yet, but Luiz is starting to). Starting with the extraordinarily difficult case-halves, and continuing with the inner plastic clips that retain the electronics module, this plastic was mis-designed and you have to bend it TOO MUCH to get the molded plastic ratchets to spring away.

In these situations, get out a fresh X-Acto blade, and trim away just .010 or .020 of that plastic. Do it manually. THEN, the ratchet action comes apart easier, for future service. Don't get carried away or you wreck it. But a little plastic removed can make for easier disassembly w/o damaging it.

If we don't get into doing some of that, the clips will simply break off in future servicings. I know because I'm already seeing bust-offs of these clips inside.

Try it sometime and you will be well-pleased.

These tips will hopefully keep your unit running happily even if it is a solderless unit.

As to battery power.... personally I recommend feeding it with AA Alkaline cells, and just dispose of them when they run down. At 38 cents a battery, from a competitive source, it's a small price to keep such a nice unit running. Others may wish to keep employing the charger. Still another option is Photo Lithium disposable batteries. However, the "Energizer" version of that has no more mA*Hr than the AA Alkaline (either is 2900 mA*Hr). Maybe another brand has truly high capacity.

When it comes to HP-34C, enjoy the most cosmetically charming little portable that HP ever made, and take good care of it, after all, eventually I plan to own all of them.

:o)

Regards,

- Norm

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

HP16C Trigonometric Functions (no inverses so far)

Posted by [Vieira, Luiz C. \(Brazil\)](#) on 13 May 2003, 1:46 a.m.

I would like to mention the fact that the programs listed here were based on the excellent Viktor Toth's Trig program written for the HP12C. There are other excellent works, including Valentin Albillo's related Trigs for the HP12C (Tried and Tricky Trigonometrics), but I was a lot intrigued with the fact that Viktor Toth used neither $n!$ nor y^x to compute each element of the series. So, I decided to go further and I literally "dismantled" the program in its stack and registers components, step by step for two loops. And there they were: $n!$ and y^x in a consecutive multiplication "fashion". Amazing! I only saw this sort of "RPN engine" in another program written for the HP41, already mentioned here, to compute the roots of a second degree equation. I ported it to the HP11C sometime ago.

I thought the program would fit fine in the HP16C because the HP16C has neither $n!$ nor y^x . I just forgot that the HP16C has no storage arithmetic, and Viktor Toth's Trigs use it a lot ($STO + n$, $STO \times n$, etc.). After a few hours refining and shrinking I could reduce Viktor's routine to a single storage register, no arithmetic required, and some additional stack manipulation. Also one small subroutine was added as I managed to group a small five-steps repeated sequence. As the HP16C has labels and the subroutine demands a RTN (return) the original SIN, COS and TAN computing needed 39 steps against the 32 used in the HP12C. As the program uses only R0, there is a final 7-byte credit.

I did not inspect the inverses so far, but I promise I'll do it soon.

I'd like to thank Viktor Toth for the inspiring program, and Valentin Albillo for showing other possibilities exist.

Restrictions applied here are the same ones applied to Viktor's Trigs, and I repeat them here:

The program fails for arguments equal to zero; For arguments outside the range of $-\pi/2 < x < \pi/2$, resulting cosine and tangent sign may not be correct; Due to rounding effects, accuracy may be less than desirable for arguments with a large absolute value ($> \pi$) and for arguments near $n \cdot \pi/2$ ($n = \text{integer}$),

The Listing

```
001 g LBL F
002 ENTER
003 ENTER
004 STO 0
005 g 1/x
006 ENTER
007 g LBL 0
008 GSB 2
009 g R^
010 GSB 2
011 RV
012 -
013 RCL 0
014 g x=y
015 GT0 1
016 RV
```

```
017   STO 0
018 g LSTx
019   CHS
020   RV
021   RV
022   GT0 0
023 g LBL 2
024   +
025 g LSTx
026   RV
027   ÷
028 g LSTx
029 g RTN
030 g LBL 1
031   ENTER
032   ÷
033 g LSTx
034   ENTER
035   ×
036   -
037 g √ (sqrt)
038   ÷
039 g LSTx
040 g RTN
```

Usage[

The calculator MUST be in FLOATing point mode:

```
[f][FLOAT] 4
[BSP]      (to clear display contents)
[g][CF] 5   (if flag 5, G annunciator, is set; not strictly necessary)
```

Given input angle (radians) x in X-register and pressing [GSB][F] we have:

Stack	Value
T	irrelevant
Z	sin(x)
Y	tan(x)
X	cos(x)

I'd appreciate any comments, suggestions, corrections, etc. The program runs correctly in my HP16C, but there may be errors after typing it in.

Thanks.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Replacement Nicads for HP Battery Packs

Posted by [Katie Wasserman](#) on 4 Mar 2003, 5:55 p.m.

Here's a list of what I use for replacement nicad batteries in all the rechargeable HP calculator battery packs. Aside from the 41C rechargeable battery pack (82120A), all of these batteries are available from Digikey <http://www.digikey.com>. What's especially nice about buying from them is that they are excellent quality batteries, are always in stock and they will pre-configure any arrangement needed for only a small extra charge. By pre-configure, I mean they will spot-weld the solder tabs together in the configuration that is needed inside the HP battery pack, thus making the battery replacement much easier and exactly the same as HP did it.

Please note: DIGIKEY DOES NOT RE-BUILD BATTERY PACKS they will only sell you the batteries inside the pack (correctly configured), you have to open the plastic pack yourself, remove the old batteries and install the new ones – this requires soldering to the ends of the pre-configured batteries for the following packs: 82001B, 82033A, 82430A. Don't send your battery packs to Digikey, just order the batteries from them using the part numbers below.

These part numbers are all for nicad batteries. Digikey does sell nimh batteries in equivalent configurations but the HP charging circuits were not designed for them, so I have not provided part numbers for those. You can try them on your own if you like by looking up the appropriate part number on Digikey's site.

Calculator(s)	Battery	Construction	DigiKey Part	Price/pack (a/o 06/2005)
Classic series HP-75C/D	82001B	3 x AA 700mah	P092-F031	\$ 8.55
Woodstock series	82019B	2 x AA 700mah (individual cells)	2 x P092	\$ 4.42
Sting series	82052A	2 x AA 700mah 2 x AA 700mah (individual cells)	P092-F021 2 x P092	\$ 5.71 \$ 4.42 ----- \$10.13
Spice series	82109A	2 x AA 700mah	P092-F021	\$ 5.71
Topcat series HP 82143A printer HP 82161A drive HP 82162A printer	82033A	4 x subC 1500mah	P175-L022	\$10.06
HP-94D/E/F	82430A	4 x 2/3C 1000mah	P238-L041	\$29.28
41C/CV/CX rechargeable	82120A	4 x 1/3AAA 50mah		\$14.00

] (see note below for details)

Notes on the above:

- All of the prices are for single quantities. And all of these batteries have a somewhat greater capacity (about 15%) than HP's original packs.
- Individual cells (with tabs, e.g. P092T) and the end cells used in packs come with solder tabs on them. For the 82019B, 82109A and 82052A battery packs remove the end tabs before inserting into the plastic pack. This will leave a flat end to the battery that matches the original HP battery pack's design. Part number P092 (no T suffix) are flat-top cells without tabs.
- There are two variants of the 82033A battery pack. In one case the V-connectors attached to the end cells inside the pack. This is how the Digikey battery pack comes and will be a cinch to install. The other variant has the V-connectors attached in the middle of the pack and will require you to do some soldering work to get the Digikey pack to match the one being replaced.
- The 82052A battery pack needs to have the end two batteries freely movable within the pack in order to make good contact with the connectors inside the calculator, so separate cells are used. The middle two cells can be joined together which helps to create more pressure on the outer two batteries for better contact.
- I have rebuilt an 82430A battery pack, a very hard task! It uses 4 2/3C cells that must be removed by cutting the pack apart. There is no seam to cut on so you need to remove the rubber grip, and cut out the entire center section of the pack (a Dremel tool with a small rotory saw blade works well for this). The 4 2/3C cells are a very tight fit, so you might want to use a smaller size (4 AA's will fit). After you replace the cells and glue the removed center section back in place, you can use a large piece of double face tape to hold the rubber grip in place. I found that carpet tape is perfect for this
- The 82120A (41C) rechargeable battery pack uses 4 1/2-N cells. Unfortunately this size is no longer made by any manufacturer. Sanyo was the last to manufacture these but they are longer available. (TG Works carried them for a while, but they are no longer in business). However, I found a reasonable substitute also made by Sanyo; 4 x 1/3 AAA cells fit with room to spare. They are slightly smaller in diameter than the 1/2 N cells and therefore have about 30% less capacity, but they work fine. They are available as part number 1/3AAA-50WT from any Sanyo distributor. The lowest price that I've found is \$2.50 each from The NiCad Lady http://nicdlady.com/cell_pr.html#nicads

Edited: 1 July 2005, 1:56 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

Creating HP41 Double X-Memory Modules

Posted by [Christoph Klug](#) on 3 Mar 2003, 5:55 a.m.

Dear HP41 enthusiasts,

follow the link www.hpmuseum.org/guest/klug/dblxmem.pdf and you will find a detailed description about creating your own Double X-memory modules for HP41.

The paper is identical to Chapter XII from the 300 page I/O-Board manual. If you want to learn more about hardware interfacing and about advanced measurement applications with the powerfully HP41 and HP-IL system, read the complete book and the new 150 page IL2000 System manual. Also you find additional information about burning Zeprom Modules and creating an extern interrupt circuit for HP41 (activating HP41 by external event).

Both books continues the basic work of Gary Friedman “Control the world with HP-IL”. Gary Friedman and other well known handheld computer authors add some prefaces to the IL2000 System manual.

Best wishes from Germany – Christoph Klug

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Flatten vintage wrinkled HP card holders

Posted by [Mark Hoskins](#) on 28 Feb 2003, 12:41 p.m.

Getting the HP 40-Card Holders Warps and Wrinkles Out

Perhaps this technique is not at the top of your list of HP Calculator repairs, but it's something that has been helpful for me. Some of the old card pacs could damage the magnetic cards if they warp or wrinkle too much causing bends in the cards. It's also then easier to use the cards as they slip in and out smoothly.

This technique works great for moderately wrinkled plastic magnetic card holders. I use an iron and three or four 5x7 cards to straighten things out.

Getting the Holder Ready... Place blank individual cards into the holder- magnetic side facing out, or reverse of what you would normally place them. They seem to slip out easier for me after they have been heated. Fill every slot. Be sure to push them all the way to the end of the holder slots. Remove any little bits of dirt or whatever that may become burned-in to the plastic.

Apply the Heat... Get a flat thick old software book preferably that has a smooth back cover when pressed down by the iron will give you a flat sturdy surface (maybe a little over an inch thick). Start out by covering the backside of one of the holder slotted pages with two index cards and position it out onto the book. Be careful about ink writing on either the book or the index cards, keep ink away from the plastic as it can transfer onto the plastic. Set the iron heat a little lower than the cotton setting. Press hard on the plastic for a short time. Sometimes I overlap the lower third with three index cards.

Apply the Heat to the Backside... Move the iron firmly down and concentrate the heat evenly. By your motions of moving the iron apply less heat to the lower third of the page. That is the portion of the page that gets the most heat and tends to over melt the pages. **ONLY APPLY THE HEAT TO THE BACK SIDE FOR ABOUT 8 SECONDS.** That's pretty fast. The task here is to warm things up for the holder front-side heating session. You will also notice that the book will begin to get hot as you process more and more pages.

Apply the heat to the Clear pocketed Front Side... This part is important to the final ability to remove the inserted blank cards. You'll be able to wiggle them out, it will just take longer if you over heat the page. This side is also quick at about 10 seconds. Move the iron back and forth minimizing the heat to the lower third again.

I like to cool down the page quickly after heating it with the palm of my hand. It's not all that hot. You can also rub out any remaining wrinkles with your hands that might have persisted.

Cover Page... This page is frustrating, as it seems to re-wrinkle after heating it, or warps a bit. Apply only the slightest heat to this one maybe 6 seconds and then repeat. Rub it out just after heating. Pull down on the open slot at the spine to get it even.

Hope this approach works for you as well as it has for me. Let me know of any improvements you might have discovered at waterhosko@aol.com. Best Regards, Mark Hoskins

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Program listers for HP-12C, 15C, 16C

Posted by [Peter Heberer](#) on 24 Feb 2003, 9:46 a.m.

Hello,

I'd like to contribute a little helper application - a program lister for the model 12C, 15C and 16C.

I made it as Excel workbooks (one for each model).

The formatting might not be the best, and maybe there's still some errors in the recognized keys - if you should find errors, let me know.

Here's where you get them:

[HP-12C program lister \(Excel workbook\)](#)

[HP-15C program lister \(Excel workbook\)](#)

[HP-16C program lister \(Excel workbook\)](#)

Here's how to use them:

Each Excel workbook consists of three sheets:

- Program: To enter your code and print it later
- Trans: code translation table (described later)
- Help: a short help text on how to use

Set your calculator to program mode (P/R) and position to the first line of your program.

Open the "Program" worksheet.

If you can't see columns B-E, click the "toggle B-E" button in column A4. The purpose is to hide some rows for later printing.

Now type in the up to three key codes listed into columns B,C,D of the "Program" worksheet.

If you type in a code combination that is not known

(either a typo, or I forgot to add it to the translation table),

you'll get a "#NV" display.

In that case, check the codes you entered, and if your absolutely sure it's NOT a typo, contact me to fix the bug.

The formatting of those columns should be explicitly "TEXT". To be sure the lookup really works, it help to key in the codes with an introducing apostroph ('), which makes Excel recognize it as TEXT, not numbers.

In column E (kLookup), the lookup code used by the sheet is listed. This is mainly for testing purpose, it's not really needed.

In column F (ListKey), the keys should be listed as the calculator lists it in programming mode.

In column G (Mnemo), a representation of the key/function is shown.

In column H (Lbl), in case of a LBL statement, the statement will be repeated in a light green background color. This makes it easier to locate a certain label in the listing.
** (The HP-12C doesn't have labels, so this is missing in the 12C lister.)

In column I (GSB), in case of a GSB statement, a "<->" symbol will be shown in a light yellow background color. This makes it easier to locate subroutine calls in the listing.
** (The HP-12C doesn't have GSB's, so this is missing in the 12C lister.)

In column J (GTO), in case of a GTO statement, a "-->" symbol will be shown in a light orange background color. This makes it easier to locate program branches in the listing.
** (column H in the 12C lister.)

In column K (comments), enter your specific comments.
** (column I in the 12C lister.)

To print, click on the "toggle B-E" button on top of the page. This hides or unhides the columns B to E. With those columns hidden, use the print preview function of Excel and see if the border/page settings need adjustment.

Some remarks:

The HP-12C has only 99 program steps. When the program memory is

cleared, all instructions are set to the command "goto line 00".
This behaviour is NOT copied by the lister worksheet.

I couldn't find a symbol for the statistical SUM sign, so I used
the letter "Z" as a printable representation.

I provided a sample program in the HP-15C lister.
It calculates the biorhythm values (physical, emotional,
intellectual) for a given birthday and a "forecast" month + year.

For each day of the month (all months have 31 days ;-)
it displays those three values which can vary from -0.99 to +0.99.
No support for that software, and no guarantee ;-)

OK, these instructions are already lengthy enough. I'll answer
further questions as they show up.
Let me know if you should find that thingy useful.

Oh - the basic trick:

The entered key codes are first "normalized", then compared to
the values in the table on sheet "Trans". The printable
representation is then displayed beside the entered key code(s).

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

HP28C: Hardware I/O basics

Posted by [Christoph Giesselink](#) on 20 Jan 2003, 3:30 p.m.

Content:

1. Fundamentals
2. HP28C memory layout
3. MMU initialization sequence
4. Master chip control registers
5. Slave chip control registers (only relevant)
6. Master display driver/timer registers
7. Slave display driver/timer registers
8. Display (used addresses)
9. Annunciators
10. Special considerations to master/slave operation

1. Fundamentals

The HP28C use 5 IC's: 1x Saturn 1LK7 CPU, 2x Centipede 1LP2 chips, 2x 64KB 1LP4 ROM chips.

The Centipede chips work in master/slave configuration.

Each Centipede chip contain

- 1KB of RAM
- display driver / timer
- I/O ports
- control registers
- power supply generator and regulator
- MMU for the internal RAM and register configuration

the MMU of each chip control the base address of the 1KB RAM and the base address of the display/timer/register area. Having two Centipede chips the total amount of user RAM is 2KB and enough drivers for a 137x32 pixel display. The display memory is placed in each Centipede chip and is separated from the user memory. Because not all display memory addresses (#40040-#40078) are wired with the display, these addresses are used as normal RAM to save some nibbles in the regular RAM area.

2. HP28C memory layout

The following memory layout is the default configuration after startup. It can be modified by the standard CONFIG/UNCNFG Saturn commands.

```
#00000 - #3FFFF 128KB ROM
#40000 - #403FF slave display driver / timer / control registers
#40400 - #407FF master display driver / timer / control registers
#4F000 - #4F7FF slave 1KB RAM
#4F800 - #4FFFF master 1KB RAM
```

3. MMU initialization sequence

```
Slave chip RAM
Slave chip display / timer / control registers
Master chip RAM
Master chip display / timer / control registers
```

4. Master chip control registers

```
#40700 LBI_CTL          LBI control
#40701 CONTRAST        Display contrast control
#40702 DSPMODE         Display mode
#40703 DSPCTL          Display control
#4070A TIMER_CTL       Timer control
#4070B RAMTST          RAM test
#4070C INPORT          RX-LED pin state and service request / type
#4070D LEDOUT          Different modes for LED output
```

5. Slave chip control registers (only relevant)

```
#40302 DSPMODE         Display mode
#40303 DSPCTL          Display control
#4030A TIMER_CTL       Timer control
```

6. Master display driver/timer registers

```
#40400 - #405DF        Display column bit map of master
#405E0 - #406DF        Display row bit map
#407F8 - #407FF        Timer (32 bit, LSB first)
```

7. Slave display driver/timer registers

```
#40078 - #402DF        Display column bit map of slave
#403F8 - #403FF        Timer (32 bit, LSB first)
```

8. Display (used addresses)

```
#40078 - #402DF        Left display area
#40400 - #405DF        Right display area
```

9. Annunciators

```
#40000 - #40007        Busy
#40008 - #4000F        Alpha
#40010 - #40017        Battery
#40018 - #4001F        Shift
#40020 - #40027        2*Pi
#40028 - #4002F        Halt
#40030 - #40037        Printer
#40038 - #4003F        All Annunciators
```

The contrast depends on the number of set bits, the more bits are set, the darker the annunciator is getting. The bits have no weight. At about 20 set bits you reach the darkest possible value.

Each bit of an annunciator address is XORed with the corresponding bit of "All Annunciators" to get the final number

of set bits.

10. Special considerations to master/slave operation

"Master" mean: Register in master controller

"Slave" mean: Register in slave controller

Timer

The slave timer only run when the master timer is running. So switching off the master timer will also stops the slave timer.

Usage in the HP28C

Master Timer (8192Hz): tick counter
Slave Timer (8192Hz): cursor blinking

The master timer is normally always running. Master Timer in connection with a 12 nibble value starting at #4F003 in user RAM builds a 1088 year counter (get the value with: # 123E SYSEVAL (1BB ROM) or # 1266 SYSEVAL (1CC ROM)). The slave timer is normally stopped. At input mode with blinking cursor the slave Timer is used to generate the blink frequency.

Christoph Giesselink
c dot giesselink at gmx dot de

Edited: 2 Sept 2003, 4:58 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

HP41 - PC Gateway & Emulator

Posted by [Christoph Klug](#) on 7 Jan 2003, 5:44 a.m.

Those HP-41 enthusiasts using the HP-IL/PC Interface Card have an excellent device for adapting the handheld computer to the modern PC world. You will find many details about interfacing the HP-41 to a PC in the I/O-Board manual, chapter XV and in HPCC Datafile Journal V16N5, V16N6, V18N3, V19N3 and V19N4.

New HP-IL/PC interface hardware now available

There is continuous demand from HP-41 users for the HP-IL/PC Interface Card which is only rarely available second-hand from sources such as eBay. Consequently, I have created a second generation, specially built version of the HP82973 ISA bus card (half-length). Small layout bugs in the first generation, special build version have been fixed and the boards have an absolutely professional design including a solder top mask and gold plated terminals. The ICs are mounted with high quality sockets, and the system includes the hard to find, exotic components IL-Terminal, IL-Transformer and IL-Chip!

Some HP-41 enthusiasts have asked about a HP-IL/PCI bus interface card version, but, for now, the design is the classic 16 bit ISA bus plug in card. Several modern PC motherboards provide one or two ISA bus slots in addition to PCI bus connections while other PCs, like the HP-Vectra, have special ISA bus extension boards, mounted beside the PCI bus motherboard. This is helpful because other specialist measurement cards and interface systems are also not available for the PCI bus. You may be concerned that modern operating systems like WIN2000/NT/XP have addressing limitations with ISA cards but, fortunately, advanced software driver tools exist which can solve these problems.

Unfortunately, the BIOS and firmware of modern PC floppy drives have limitations working with LIF disks created by the HP9114 IL-Disk-Drive. Solving this is only possible by using older PC hardware, for example a nostalgic 286/16MHz PC, and which is also compatible with the unique LINK PLUS transfer software package for HP-41, HP-71 and HP-75 handheld computer.

Altogether the classic ISA bus version is the best compromise and different reasons exist for HP-41 users to install the HP-IL/PC Interface Card in an old PC or in a modern (ISA bus) compatible PC!

HP-41 ↔ PC gateway software

Beyond the basic idea of creating a second generation HP-IL/PC Interface Card, there was a vision of having advanced PC software enabling many more gateway features between the HP-41 and PC worlds. With the original card, users needed to use a combination of software tools such as TRANS41 and additional HP-41 emulator software packages. What is wanted is a powerful "all in one" solution which includes gateway/transfer features plus advanced HP-41 emulator capabilities!

Therefore I contacted Jean-Francois Garnier, asking him about producing a new and expanded version of his HP-41 PC emulator software EMU41. The real HP-41 is much more than a mere calculator, it can also be a system controller using the HP-IL Interface Loop. With this some surprising computer setups are possible including hardware

interfacing, measurement and control applications using an I/O-Board or IL2000 System. Jean-Francois is mirroring these system features from the real world in his virtual EMU41 and he is now the first to include HP-IL support in an emulator!

With the new EMU41, the HP-IL/PC Interface Card acts as gateway to the real world. It can control real external IL-Devices, for example reading the HP3468 IL-Digital Multimeter and outputting data to the HP7470 IL-Plotter! Additionally, the new EMU41 can provide and control some virtual IL devices like DISPLAY (video screen), PRINTER (routes data to LPT1 parallel port), FDRIVE1 (working with HP9114 LIF disks), HDRIVE1 (a new mass storage unit) and DOSLINK (for ASCII data transfer to PC world).

Real HP-41 and virtual EMU41

Furthermore Jean-Francois brings both together, the real HP-41 and the virtual EMU41. Connect the "forty one" to the HP-IL/PC Interface Card and switch on the calculator: The handheld scans the Interface Loop and detects the virtual IL-Devices from EMU41. Select the HDRIVE1 mass storage device for easy uploading and downloading of data and programs between the HP-41 and EMU41. Transfer the HDRIVE1.DAT file by email and you get fast and easy exchange of HP-41 data, programs and ROM image-files with other HP-41 enthusiasts. Select the virtual DOSLINK device for uploading ASCII data from real HP-41 to the PC world for post-processing with a spreadsheet, for example.

For advanced data logger applications with the HP-41, you can also transfer non-normalised numbers (NNNs) from the handheld to EMU41. EMU41 decodes the received numbers and stores them to a DOS file for data import to other PC software applications.

Last but not least, EMU41 is truly an advanced HP-41 emulator. By installing ROM image-files, EMU41 can create fast HP-41 Barcode plots (SKWIDBC-ROM plus PRINTER) and gives you real HEPAX-Module or RAM-BOX features (MLDL.DAT file plus ZENROM or W&W or ERAMCO operating system).

A nice double pack for HP-41 enthusiasts

The vision has become reality and EMU41 has grown into the necessary "all in one solution" – the software tools LINK PLUS or TRANS41 are no longer needed (but also compatible with the new card). The nice hardware/software combination of HP-IL/PC Interface Card and EMU41 gives you useful gateway and advanced emulator features. Future extensions of EMU41 will following later, for example supporting a second virtual HDRIVE2.

Making this nice double pack available to HP-41 enthusiasts is possible because of the helpful inputs of Emmanuel Compes and Wlodek Mier-Jedrzejowicz (adding some of the exotic IL-Transformers and IL-Chips), Leo Duran (testing an address routing software tool for WIN2000/NT/XP) and the ingenious work of Jean-Francois Garnier (expanding his nice EMU41 with powerful HP-IL features)!

The new EMU41 software will be available by the end of November 2002. For the full version (including HP-IL) Jean-Francois wants a fee of \$20. The price for a new, function tested HP-IL/PC Interface Card is \$150. Remember, this version is a hand-built edition and not comparable with usual mass marked products. For more information about this powerful HP-41 & PC gateway solution contact jfgarnier@wanadoo.fr for ordering a personal signed full version of EMU41 or the author christoph.klug@de.bosch.com for getting the new HP-IL/PC Interface Card including some accessories.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Hiding Files in HP48G series calculator

Posted by [HPunk®](#) on 5 Jan 2003, 11:12 a.m.

There is a hidden directory, at the HOME directory, which is called " (empty name, null character(NOT TO CONFUSE with CHR 0)). The only way to get it is #15777h SYSEVAL(SYSRPL command NULLID). This program will do all procedure: « HOME #15777h SYSEVAL EVAL ». After an ON+A+F you will see 3 files, that SHOULDN'T be ever touched. But you can store anything you want into that directory; it will be hard to be found.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

Solving the problem of "HP49 Crashes when installing libraries in Flash ROM (Port 2)"

Posted by [HPunk®](#) on 15 Dec 2002, 8:21 a.m.

The HP49, mainly the emulator, may lock when installing libraries in port2, the only way to get out is the hard reset, and this causes a "Try To Recover Memory?".

The solution is fairly simple: 1 - Download the library and recall it to stack. 2 - Install it in PORT 1. 3 - Do an ON+C 4 - Go to filer application 5 - Move from port 1 to port 2. If it locks during, you may have to do an hard reset, but this time it won't do a TTRM, just an Invalid card data, that is cured by an OFF-ON and the libraries will not be moved.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

Opening a Voyager

Posted by [Steve](#) on 29 Nov 2002, 4:24 p.m.

First, thanks to everyone who provided their thoughts and experiences about opening/repairing a Voyager series HP. Here's what I found:

The brushed aluminum bezel is easily deformed if you attempt to remove it. I quickly gave up on this approach after the first crease appeared.

I took Raymond's advice (thank you!) and removed all the heat stakes except the bottom row. This allowed me to clean the display without totally removing the PCB. As you swing the PCB out the keys will move around and it's time consuming to move them back into their correct slot.

It took a long time to clean the inside of the display window because there seems to be some coating that flakes off as you clean it. It almost looks like a car window partly covered with ice. I had to clean the window until all of the coating was removed to have it look clear.

Use a razor blade (or Xacto knife) to remove the heat stakes and save them. After the repair is complete you can melt them (I only did a few in evenly distributed locations) with a soldering iron back into place. As you melt them, push the PCB down to get a tight seal. I'm not sure how strong they are, but they seem to hold.

I added a couple of 3M clear plastic rubber bumpers (self stick) to the bottom of the case which will help support the PCB. These, plus the full bottom row of stakes, 8 or so re-melted stakes, and the battery clip, seem to give adequate support for the keyboard.

In conclusion, leave the inside of the display alone! It turns out the "dust" particle I was originally trying to remove is a defect in the display window plastic. By inserting a thin piece of laminated paper, I disturbed some sticky material and caused the problem. I'm basically back where I started, with a weaker PCB, and a small crease in the bezel (not really noticeable, but it stands out to me!). If you can't clean it with compressed air, leave it alone!

I hope this helps others down the road. Happy calculating,

Steve

Edit Message

Delete Message

Password:

[\[Return to the Message Index \]](#)



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

ElectroStatic Discharge: a possible solution for the HP48

Posted by [Vieira, Luiz C. \(Brazil\)](#) on 27 Nov 2002, 7:25 a.m.

Should you experience unexpected behavior in your HP48 and suspect it is related to ESD (ElectroStatic Discharge), there is a keystroke procedure that will allow you to check for it and return the calculator to its normal operation. This keystroke sequence have been used and solved problems probably related to ESD, but there is no concrete evidence it is an internal discharge "hardware". The sequence to enter in the ESD screen is as follows:

switch the calculator to ON;
press and hold [ON] key;
press and hold [D] key;
release [ON];
release [D];

The calculator enters in interactive test mode. Now press [MTH] key and the display shows:

```
ESD 210V 210V
   ||    ||
```

The calculator will beep for a while; see if there is any warning message. To get rid of the test screen, do it:
press and hold [ON] key;
press and hold [C] key;
release [ON];
release [C];

Now try the calculator again. If the problem is related to ESD, it is probably solved.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Removing the back labels to HP classic series calculators

Posted by [Mark Hoskins](#) on 10 Nov 2002, 11:53 p.m.

Here is a method to removing the back label to the Classic HP 35,45,55,65,67 and 80. If followed carefully you should be able to replace them with near perfect appearance. This approach has worked well for me.

Step 1 Your tools

a) Get a very sharp Exacto knife with a crescent blade (curved in a semi circle). Do not use the very pointed blade because it can cut the label. Have the blade mounted to a nice big handle. b) Get a hairdryer just for this purpose. They are \$3 at the Goodwill or any secondhand store. I use a 1600 BTU rated standard sized hairdryer. Use it on high unless it is the type that generates a very strong flow. You want a steady fairly fast air flow but not too hot so that it melts the back battery plate of the calculator (not fun at all).

Step 2 Hand-towel Wrap

Remove any batteries from the calc and replace the door to help seal the calculator from the hot air. Sit down and wrap the calc in a hand towel, except for the label, take a deep breath (really deep if it's an HP-65). Place the calc label up with the upper portion wrapped in a doubled over towel. Place everything on your legs as you sit on a chair. Slightly overlap (1/4") the top of the label with the towel so the hot air will not partially melt the case or back door.

Step 3 Turn on the Heat

Turn on the hairdryer and aim it away from the battery compartment. Move it back and forth to evenly heat the label. Use the tips of your fingers on a quick touch across the label to feel the heat of the label build up. Hold the dryer about 2-2.5 inches above the label and move it back and forth to evenly heat up the label surface. Heat it depending on how you feel the label's heat, about 30-60 seconds.

Step 4 Lower left label corner

It is not clear why, but the lower left corner seems to be the best and most "pull-able" spot on the label to start the removal. Once in a blue moon the lower right also works. In a ticking motion, use the curved Exacto knife to lift up enough (about 3/16 inch) to pull with your fingers. Just before pulling drop the hairdryer like a hot potato. Concentrate on pulling up on the label at first, then pull more in tension evenly after the left edge rises (not just an upward motion). The label will curl slightly, no problem the hard part is over.

Step 5 Label smoothing

Take a piece of the backing of the printer labels that is smooth and gently push the sticky side of the HP calculator

label onto it. Overlap the paper part by about ½ inch. With you fingers generally straighten the edges and the curved part. Ideally you want to have a slight outward bend to the top of the label when finished. Get a smooth bottom spoon and place the HP label face down on your kitchen counter. Rub the paper side of the label and back of the label with the spoon (spoon rounded bottom touching the paper). I tried a plastic spoon and that worked well recently.

Start with the edges and flatten them then work your way into the center of the label. It is amazing how this flattens it all out. Experiment with one or two pieces of paper on the kitchen counter to see if that helps. It appears best to have the Hp label printing touching the kitchen counter. Make sure the counter is very smooth and clean before letting the label touch it.

Step 6 Resetting the label

Peel back the paper to reveal the sticky side of the HP label. Carefully clean the back of the calculator; leave no residue of the glue that may have stayed on the calculator. Align the TOP OF THE HP LABEL to reset it. Aligning the left side does not always work well and misalignments have happened to me.

That is how I remove these labels, if anyone has a better way I'd love to know. There is an HP collector in Germany who has another method he says is failsafe. He would have told me how if I won his bid, but I missed it. He said it took 3 hours, so that implies to me that he soaked it in some solvent. That may be possible if you have a very shallow dish that only the label would be immersed.

Any comments or other approaches are gladly received.

Best Regards,
Mark Hoskins
(Waterhosko@aol.com)

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

True Type Fonts® for the HP41, Voyagers and 82143A

Posted by [Vieira, Luiz C. \(Brazil\)](#) on 25 Aug 2002, 4:16 p.m.

In 1995 I wrote a book for both HP41 and HP48 calculators. At that time I missed a TTF that would simulate these calculators' LCD characters and final printing in the HP82143A (also 82240A/B). I could not find any TTF that was closed to what I wanted.

In just a few words: I developed the TTF's after the book was in the bookstores, and I want to share them with you now. Dave is hosting the files and they can be downloaded.

[Voyager-series](#) LCD character set (*)

[HP41's](#) LCD character set

[82143A](#) Printer Listings

If you have any suggestions, corrections or general comments, please e-mail me.

Best regards.

(*) This TTF has many characters not available at the Voyager's LCD. I added them for personal reasons and I thought about removing them, but I decided to keep them there so other users would access them to. They are lower- and upper-case lettering, and can be used for titles or any other special warning message (not suitable for direct text).

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

99 Digits of PI on an HP 32SII

Posted by [Katie Wasserman](#) on 19 June 2002, 11:08 p.m.

(See the more recent article on this forum for a simpler algorithm then produces 200 digits of pi on the 32sii.)

Just for fun, I tried to see how many digits of pi I could get a 32SII to calculate in a reasonable amount of time. This program computes 99 of them (only the first 96 of them are correct) in around 11 minutes. It makes heavy use of store and recall register arithmetic to save programming space. It also makes use of 12 digit accuracy, but it could be adapted to most any other HP calculator with enough memory and indirect addressing.

After entering the program and verifying that the checksums are correct, you can run it by just using XEQ P. Wait awhile (11 minutes for me) and the program will end. The result will be in registers A-K, 9 digits in each register (no decimal points).

First 99 digits of pi (first 96 are correct) based on the following formula:

$$\text{Pi} = 20 * \arctan(1/7) + 8 * \arctan(3/79)$$

And using:

$$\text{Arctan}(x) = (y/x) * (1 + y^2/(3) + y^4/(3^3) + y^6/(3^5) + y^8/(3^7) + \dots)$$

$$\text{where } y = x^2 / (1 + x^2)$$

The initial terms of these expressions are built into the program as 2.8 and 0.30336

$$\text{Pi} = \text{term}_1 + \text{term}_2$$

$$\begin{aligned} \text{term}_1 &= 20 * \arctan(1/7); \text{ so } x=1/7; y=1/50 \\ &= 2.8 * (2/3*y + 2/3^3*y^3 + \dots) \end{aligned}$$

$$\begin{aligned} \text{term}_2 &= 8 * \arctan(3/79); \text{ so } x=3/79 \text{ and } y=9/6250 = (9/125)/50 \\ &= .30336 * (2/3*y + 2/3^3*y^3 + \dots) \end{aligned}$$

```

-----
LBL P                                start of program
CLRVARs                             result will be in registers A-K, L-V is the partial sum
                                     X, Y and I are also used
7
10^X
28
X                                     low memory way of getting 280,000,000; starting value
STO L
XEQ A                               add to result register
2

```

LBL Q	STO Y	starting loop value
	XEQ X	start of 1st term loop
	X>0?	do main part of loop
	GTO Q	if X is not zero then the partial sum still needs to be reduced
	30,336,000	stating value for 2nd term
	STO L	(note: M-V will be zero when 1st loop loop ends)
	XEQ A	add to result register
	2	
LBL R	STO Y	starting loop value
	9	start of 2nd term loop
	XEQ M	multiply partial sum by 9
	125	
	XEQ D	divide by 125
	XEQ X	do main part of loop
	X>0?	If X is not zero then partial sum still needs to be reduced
	GTO R	
	STOP	done, result is in A-K
		(9 digits/register, leading zero's won't display)
LBL X		subroutine to do one step of 1st or 2nd term
	RCL Y	get loop counter
	XEQ M	multiply partial sum by it
	1	
	RCL+ Y	add one
	XEQ D	divide by it
	50	
	XEQ D	divide by 50
	XEQ A	add partial sum to result
	2	
	STO+ Y	add 2 to loop counter
	RCL V	get low order 9 digits of partial sum
	RCL+ L	add in low order 9 digits and return it
		(this is a heuristic,
		if X is zero then we assume partial sum is all zero)
	RTN	

P=15.0 CK=1038
 Q=21.5 CK=810E
 R=13.5 CK=AF06
 X=21.0 CK=3B57

LBL A		Start of subroutine to add partial sum into result
	11	
	STO i	starting value for loop counter
	0	starting value for carry
LBL B		start of loop
	11	
	STO+ i	point to 9 digits of partial sum
	R\	drop the 11 from the stack
	RCL+(i)	add the partial sum 9 digits to the carry
	11	
	STO- i	point to 9 digits of the result
	R\	drop the 11 from the stack
	9	
	10^x	low memory way of getting 1,000,000,000
	x<>(i)	temporarily exchange 1,000,000,000 and the 9 digits of the result
	+	add the partial sum + carry to the result - just these 9 digits
	RCL/(i)	divide by 1,000,000,000
	FP	the fraction part will only be 9 digits
	STOx(i)	this will be a 9-digit integer now stored in result
	LASTx	get the result of the divide back
	IP	the integer part will be the new carry
	DSE i	decrement the loop counter and check for done
	GTO B	process next 9 digits
	RTN	

A=06.0 CK=A9D7
 B=30.0 CK=099E

```

LBL M          start of subroutine to multiply partial sum by top of stack
                save the multiplier in X
                STO X
                22.011
                set loop counter
                STO i
                0
                to start with register V and run to L
                starting value for carry
LBL N          start of loop
                RCL(i)
                get 9 digits of partial sum
                RCLX X
                multiply it by the argument saved in register X
                +
                9
                add in the carry
                10^x
                low memory way of getting 1,000,000,000
                STO (i)
                temporarily store this in place of the multiplicand
                /
                divide the product by 1,000,000,000
                FP
                the fractional part will only be 9 digits
                STOx(i)
                this will be a 9-digit integer now stored in the partial sum
                LASTX
                get the result of the divide back
                IP
                the integer part is the new carry
                DSE i
                decrement the loop counter and check for done
                GTO N
                process next 9 digits
                RTN

```

M=15.5 CK=4B87
N=22.5 CK=1F72

```

-----
LBL D          start of subroutine to divide partial sum by top of stack
                save the divisor in X
                STO X
                12.022
                set loop counter
                STO i
                0
                to start with register L and run to V
                starting value for carry
LBL E          start of loop
                STO+(i)
                add carry to 9-digit dividend
                RCL (i)
                get the sum
                RCL/ X
                divide by the argument save in register X
                IP
                the integer part will be the quotient
                X<>(i)
                save it and get the original dividend
                RCL (i)
                get the quotient on the stack
                RCLx X
                compute the remainder by multiplying by the divisor
                -
                and subtracting from the dividend
                9
                low memory way of getting 1,000,000,000
                10^x
                adjust the remainder to be the new carry
                x
                increment the loop counter and check for done
                ISG i
                GTO E
                process next 9 digits
                RTN

```

D=15.5 CK=5117
E=22.5 CK=FCD4

Edited: 5 Nov 2008, 10:41 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP28S: Hardware I/O basics

Posted by [Christoph Giesselink](#) on 19 June 2002, 12:23 p.m.

Content:

1. Fundamentals
2. HP28S memory layout
3. Master chip control registers
4. Slave chip control registers (only relevant)
5. Master display driver/timer registers
6. Slave display driver/timer registers
7. Display (used addresses)
8. Annunciators
9. Special considerations to master/slave operation

1. Fundamentals

The HP28S use 3 IC's: 2x Lewis 1LR2 chips, 1x 32KB RAM.

The Lewis chips work in master/slave configuration. Only in the master chip the CPU is running.

Each Lewis chip contain

- 64KB of ROM
- 2 external memory controller
- display driver / timer
- control registers
- MMU

the MMU and ROM is mask programmable, so the user can't change it's content. Having two Lewis chips the total amount of ROM is 128KB. The display memory is placed in each Lewis chip and is separated from the user memory in the external RAM chip.

2. HP28S memory layout

```
#00000 - #3FFFF 128KB ROM
#C0000 - #DFFFF 32KB RAM (mirrored)
#FF800 - #FFBFF slave display driver / timer
#FFB00 - #FFB0F slave control registers
#FFC00 - #FFFFF master display driver / timer
#FFF00 - #FFF0F master control registers
```

3. Master chip control registers

#FFF00	RATECTL	Rate Control (speed)
#FFF01	CONTRAST	Display contrast control
#FFF02	DSPTST	Display test
#FFF03	DSPCTL	Display control + BIN
#FFF04	CRC	Crc (16 bit, LSB first)
#FFF08	LPD	Low Power Detection
#FFF09	LPE	Low Power detection Enable
#FFF0A	MODE	Mode register
#FFF0B	RAMTST	RAM test
#FFF0C	INPORT	RX-LED pin state and service request
#FFF0D	LEDOUT	Different modes for LED output
#FFF0E	TIMER1_CTL	Timer1 control
#FFF0F	TIMER2_CTL	Timer2 control

4. Slave chip control registers (only relevant)

#FFB03	DSPCTL	Display control + BIN
#FFB0E	TIMER1_CTL	Timer1 control
#FFB0F	TIMER2_CTL	Timer2 control

5. Master display driver/timer registers

#FFC00 - #FFE5F	Display area of master
#FFFF7	Timer1 (4 bit)
#FFFF8 - #FFFFF	Timer2 (32 bit, LSB first)

6. Slave display driver/timer registers

#FF800 - #FFA5F	Display area of slave
#FFBF7	Timer1 (4 bit)
#FFBF8 - #FFBFF	Timer2 (32 bit, LSB first)

7. Display (used addresses)

#FF840 - #FFB5F	Left display area
#FFC00 - #FFE27	Right display area

8. Annunciators

#FF800 - #FF807	Busy
#FF808 - #FF80F	Alpha
#FF810 - #FF817	Battery
#FF818 - #FF81F	Shift
#FF820 - #FF827	2*Pi
#FF828 - #FF82F	Halt
#FF830 - #FF837	Printer
#FF838 - #FF83F	All Annunciators

The contrast depends on the number of set bits, the more bits are set, the darker the annunciator is getting. The bits have no weight. At about 20 set bits you reach the darkest possible value.

Each bit of an annunciator address is XORed with the corresponding bit of "All Annunciators" to get the final number of set bits.

9. Special considerations to master/slave operation

"Master" mean: Register in master controller

"Slave" mean: Register in slave controller

Display

To turn on the display each controller must be turned on separately. The order is important, you must first turn on the slave then the master controller. Turning off should be done in reverse order, first the master then the slave controller.

Timer

The slave timer only run when the master timer is running. So switching off the master timer will also stops the slave timer.

Usage in the HP28S

```
Master Timer1 (16Hz) : unused
Master Timer2 (8192Hz): tick counter
Slave Timer1 (16Hz) : unused
Slave Timer2 (8192Hz): cursor blinking
```

The master timer is normally always running. Master Timer2 in connection with a 12 nibble value starting at #C0003 in user RAM builds a 1088 year counter (get the value with: # 11CAh SYSEVAL). The slave timer is normally stopped. Only at input mode with blinking cursor slave Timer2 is used to generate the blink frequency. That both Timer1 are unused has a historical background, the predecessor of the HP28S, the HP28C has only two 8192Hz hardware counter equal to the current Timer2.

Christoph Giesselink
cgiess@swol.de

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

HP-19C printer overhaul discoveries

Posted by [Michael Meyer](#) on 27 May 2002, 12:11 a.m.

Hi to all.

I just finished a complete overhaul of my HP-19C printer. Why? Because it was there. I ended up using the best parts from 2 printers (one is a parts machine). I wanted to post my lesson so as to help others do this more easily....

1. Remove the motor. Lubricate it, if needed, with very fine oil.
2. Remove the pressure plate that pushes against the print head. I polished mine on a flat fine-toothed file. Restored a "mirror" image (well, close for plastic).
3. Remove the print-head metal bracket to expose, remove, and clean the little tiny brass bit that guides the printhead on the cross-cut motor shaft. Be careful not to lose the spring holding it in! I find it best to now gently remove the print-head itself from its bracket to clean and to not be swinging it around with the printer overhaul.
4. The motor shaft will remove after the spring clip is carefully removed from the left-hand side furthest from the motor. The shaft and guide bar then come out with a little spreading of the frame.
5. The bottom shaft and gear, there to make the paper advancing frame move evenly, isn't on a shaft. (I found out the hard way... good thing I have extras...) The gear, top and bottom paper guide assembly are held in by brass pins.
6. Remove the brass pins that hold the paper guide mechanism in place. There is a fixed top section and a moving (advance mechanism) bottom section.
7. Be careful not to lose the spring that presses the top paper guide against the printhead pressure bar.
8. Remove and clean the parts. Carefully remove the spring clips holding the printer paper advance parts together. The inner surface of both the top and the bottom contain a spring holding in a metal bar. Clean these carefully and tighten the leaf springs if necessary. These bars are what hold and advance the paper. It looks like they'll never wear out. Nice design. The guides themselves are held in parallel by two thin plastic paper guides, one on each side.
9. Reassemble unit. I found out... again, the hard way, that the upper mechanism is hard to push in place against the pressure spring and then put in the brass pins. Instead, install the bottom mechanism, then install the upper mechanism with the brass pins. The upper mechanism will swing out, allowing you to replace the pressure spring easily. Then just drop in the two paper guides (nibs "up" to lock in place) on both sides.
10. Replace the motor shaft and guide bar with printhead unit in place, and then replace the little brass part with the spring and metal clip. Don't forget the snap ring clip on the left side of the motor shaft. Replace the motor. Replace the printhead in its carrier to come just even with the top of the pressure bar. Check to see if the relay-type switch at the bottom functions well and is clean.
11. Turn the shaft to test and to move the printhead to the right. This should also drive the lower paper guide up and

down with each excursion (to advance the paper). Now, reinstall the whole print mechanism.

My printer now prints much darker (probably due to the smoothness of the pressure bar and the clean reassembly of the spring on the upper paper guide which holds the pressure bar also). The paper advance leaves uniform space now, as it grabs the paper perfectly again. Uses up more paper, but looks REALLY nice.

Let me know if I've forgotten anything. Just wanted to save everyone the trouble of finding out about the brass pins only after breaking the gear or other parts....

Peace Michael

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Calculator Fiction -- The Mutant HP

Posted by [Paul Brogger](#) on 24 May 2002, 11:47 a.m.

The following is an excerpt from a college math assignment, ca. 1978. We were supposed to demonstrate our facility with calculation and prediction of celestial motions in the form of a personal letter to our professor, who was assumed to be taking a hypothetical trip and was asking about the timing of a star's zenith arrival times during a ten-day stay at a particular location. I went off on a tangent, somewhat inspired by my HP-29C. I hope you enjoy it.

. . . that inconvenience is coupled with the fact that I've only had my slide rule available for figuring lately; my calculator having seemingly mutated to some higher order of electronic integration.

Recently, during some particularly complex calculations, the thing became insensitive to input of any sort, and began an unceasing series of apparently random manipulations, derivations, and rearrangements. I've attempted an analysis of the sequence of figures displayed, but can detect neither logical order nor numerical series to explain what has happened or to predict what will.

At first, the lazy, arrhythmic pulsation of the dull red display exerted a surprisingly soothing effect (especially after a few hits of Lebanese Blonde). Long sessions that began with close, energetic observation and slavish transcription inevitably devolved to slouching, hypnotic séances, altogether void of analytic merit. Lately, however, the beast's quasi-Algebraic ruminations have taken on a rather sinister character. Intense staccato bursts of activity are interspersed with ominous (dare I say, *moody*?) periods of inactivity, during which I can almost *hear* an obstinate, insidious concentration directed at the supra-Arabic numerals in the register.

Now it can turn itself off and on at will, and it grows perceptibly warm to the touch whenever I try to disconnect his recharge cord. I fear he'll soon learn to bypass the display, so that I won't have the vaguest clue as to what's going on *inside* . . .

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

A Common Woodstock Problem and Repair Suggestions

Posted by [katie](#) on 16 Apr 2002, 10:14 p.m.

(The following was extracted and edited from threads in the discussion forum here during August 2000 and some later discussions)

A seemingly common problem on some Woodstock machines is that they don't work until they are left on to "warm up" for some period of time. After leaving the on for several minutes (maybe up to 1/2 hour) then turning them off and back on again they start working normally. More specifically the following problems were found:

In the case of the 22, 25 and 25C, the CPU chip works fine (i.e., you can do any command that just uses the stack) but the if you do any command that accesses the external memory chip you either read a zero or a crazy number, a write will do nothing. On the 27 the whole machine is locked up until it's warm, because the RAM and ROM are both on the same external memory chip. The lock up may manifest itself in the form of a totally blank display, one with all zero's or one with a fixed "0.00".

Internally, the problem seems to stem from the clock signals being generated at double their normal frequency until the warm up and "reboot" of the calculator is done. The problem chip seems to be the ACT chip and the signals it generates but it might be with loading from the RAM/ROM chip. The ACT chip is the same in the 22, 25 and 25C, but is different in the 27, which is based on NMOS as opposed to PMOS. The 21 is also an PMOS design but I have seen no reports of the problem with it. The 25C and 29C use some CMOS technology and some NMOS, however the 29C (at least) seems to have an entirely different set of problems. The following solutions have been found to fix this problem on the 22/25/25C/27 so that the calculator works as it should initially:

There are two clock signals on the 22, 25, 25C and 27, one is approximately 175Khz the other is about 1.6KHz. On the 22, 25 and 25C the problem seems to be with the lower frequency clock signal having excess noise while on the 27 it's with the higher frequency signal being doubled.

On the 22 I found that the 175Khz-clock signal was coming through the unused pins on RAM/ROM chip (the one closest to the battery compartment and did the following fix: I simply connected one of the unused pins on the RAM chip to ground. (At least some of the unused pins appear to be connected to the substrate of the chip.) In particular, I connected a jumper between pin 7 (ground) and pin 10.

On the 25 this repair did not work, although it too had clock noise coming through on the same unused pins. However, I found that the data signal line was quite noisy too. The fix here was to install a 20K resistor between the data line and ground. The data line is pin 11 of the ACT chip (the ACT chip is the one at the bottom of the circuit board.) and there is a convenient ground just across the chip on pin 12. One fellow collector has reported that this same fix worked as well on the 25C.

On the 27 what I found was that when the RAM/ROM worked, the 1.6KHz line was really 1.6KHz. But when it was not working it was exactly twice this frequency (i.e., 3.2KHz)! I isolated the clock line and found that the ACT chip is the one that generates this clock and outputs it on pin 20. For some reason that I don't understand, depending on the load on this pin it runs at 1.6KHz or 3.2KHz. I found that if I resistively couple this clock line to the rest of the circuit I

can get it to run reliably at 1.6Khz. What I did was cut the trace from pin 20 to the rest of the circuit and inserted a 20K resistor. The calculator now runs perfectly regardless of the ambient temperature (well, within reason at any rate).

Notes on the HP 27 circuit power supply:

Hook up an external 2.5 volt power supply with an ammeter in series. If the power supply is working, you should see around a 100 ma (blank display) to 200 ma (most segments on) current draw. Assuming that you're getting that, you should next check the output voltages from the inverting supply. If you measure them relative to the negative battery terminal you should find three of them: -3.3, +6.5 and +12.5. The easiest place to find these is on the non-component side of the main circuit board on 16-pin IC pattern that doesn't have a chip installed in it (labeled U5). Pin 9 = 6.5v, Pin 12 = -3.3v, pin 15 = +12.6v.

Edited: 15 Feb 2005, 3:10 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-42S: New Facts

Posted by [J-F Garnier \(France\)](#) on 10 Apr 2002, 4:32 p.m.

Contents:

Introduction

1. ROM dump

2. Hardware

2.1 Display

2.2 Annunciators

2.3 I/O registers

2.4 Clock

2.5 Keyboard

3. Software

3.1 System RPL internals

3.2 User code entry tables

3.3 The hidden XFCN keyword

3.4 The COMP bug

Introduction

Few information is available on the hardware and software design of the HP-42S. Most useful documents are:

- The HP-42S description in the HPDATABase by C.A. Finseth at <http://www.finseth.com/~fin/hpdata/hp42s.html>
- Pioneer "Observational Internals" , by Paul Brogger, in this forum, at <http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/articles.cgi?read=5>
- Increasing HP-42s Memory to 32K, by Paul Brogger, in this forum at <http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/articles.cgi?read=199>

Please refer to these documents if needed.

In the process of developing an emulator of the HP-42S, I discovered new facts that can be also useful to expand the HP-42S. Actually, I have a very basic running emulator version (text mode DOS) that I only use to study the HP-42S internals (it is NOT suitable for publication, don't ask me, sorry!).

1. ROM dump

In order to study the HP-42S internals, or to even think about an emulator, the first thing to do is to get a ROM image. Fortunately, there is a way to do it thanks to the internal memory browser. You will need: a HP-42S (!), a HP48 (with the INPRT program from HP) and a PC.

Here is the procedure:

- enter the test mode on HP42S: EXIT-LOG

- enter the memory browser on HP42S: <- (back arrow)
- here, you can check your software version: press '.' (decimal point). Version is displayed at the upper left (A, B or C). At the upper right, you can see the last warm start log (similar to the WSLOG command on HP48 serie)
- use the ^, v (up and down arrows), /, *, +, - keys to go to address 0
- now, align your HP42S and HP48 IR leds
- start INPRT on the HP48
- press the COS key on HP42S and keep it down
- monitor the address displayed on HP42S, and release the COS key when reaching 1000h (you can release the COS key a little bit before and add a few keystrokes to stop just at 1000h)
- wait for INPRT timeout (about 1s)
- now, you have your first 1000h nibbles on the HP48! Store this string in an object, and transfer it to the PC.
- Continue up to the end of the ROM (1ffffh). I recommend to transfer by 1000h blocs for convenience, so you will need 32 transferts. Be patient, it takes a lot of time ...
- To leave the HP42S test mode, press EXIT-Sqrt

2. Hardware

By partially disassembling and tracing the code, I managed to find out some important hardware ressources.

2.1 Display

The display is made of 131 columns numbered from 0 to 130.

Each column is made of 4 consecutive nibbles (16 pixels).

The 4 consecutive nibbles drive pixels from top to bottom.

Each nibble is a column of 4 pixels. LSB=top pixel, MSB=bottom pixel

The display is organized in two interleaved areas:

```

40000 : column0
40004 : column66
40008 : column1
4000c : column67
40010 : column3
...
401f8 : column63
401fc : column129
40200 : column64
40204 : column130
40208 : column65

```

2.2 Annunciators

Each annunciator is controlled by one 5-nibble word.

Valid values seem to be either 00000 (off) or FFFFF (on)

One word at 40210 seems to control all annunciators globally. Should be 00000.

```

40210 : ^v --^ print ((.)) batt G RAD
40218 : ^v
40220 : --^
40228 : print
40230 : ((.))
40238 : batt
40240 : G
40248 : RAD

```

2.3 I/O registers

The 40300-4030f area is used for I/O registers. The usage of these locations is not completely clear for me. Note that the organization of this area is close to the HP-28S. Note also that a 4-nibble hardware CRC calculator is included.

```

00: speed
01: contrast
02

```

```

03: display?
04-07: CRC
08: battery
09
0a
0b
0c: irin??
0d: irout?
0e: beep?
0f

```

2.4 Clock

The clock is at 403f8-403ff: 8 nibbles.

The clock counts down at 8192 ticks per second.

Address at 403F7 is also related to the clock, and could be a status/control register?

2.5 Keyboard

The keyboard handler uses a circular 16 bytes buffer at 50086-500a5. The two nibbles at 50084 and 50085 hold the head and tail pointers. The one-byte codes stored in the buffer are simply the natural order of the keys from left to right and from top to bottom (just like the GETKEY command): 1 for 'Sigma+', 2 for '1/x', and so on until 37 for '+' (the Enter key counts for one key only). Shifted key codes are stored by adding 37 to the corresponding unshifted key code: from 38 for 'Sigma-' to 74 for 'CATALOG'.

3. Software

The HP-42S internal software is based on RPL, but system RPL is not accessible to the user (as far as I know). 3 software versions are known: A, B and C. The following applies only to the C version.

3.1 System RPL internals

The internal RPL uses a subset of objects used in the HP-28C/S: binary integer, real, extended real, complex, extended complex, character, program, code, global variable, local variable:

```

=DOBINT      EQU #02911
=DOREAL      EQU #02933
=DOEREAL     EQU #02955
=DOCMP       EQU #02977
=DOECMP      EQU #0299D
=DOCHAR      EQU #029BF
=DOCOL       EQU #02B3E
=DOCODE      EQU #02B6D
=DOIDNT      EQU #02BCB
=DOLAM       EQU #02C6E

```

I found some system RPL entries similar to the HP-28C/S, like:

```

=DUP         EQU #1B729
=2DUP        EQU #1B74D
=SWAP        EQU #1B76D
=DROP        EQU #1B782
=2DROP       EQU #1B796
=ROT         EQU #1B7A4
=OVER        EQU #1B7D1
=PICK        EQU #1B7E8

```

There is for sure a lot of other entries similar to the 28C/S.

3.2 User code entry tables

I managed to find the user code entry tables. My goal was to study the strange XFCN function, and also explain the well-known COMP bug.

There are 3 tables:

- main table: codes are the same than the 41C.
This table is located at 1b97d and is 500h nibbles long (256 entries, each entry is a 5-nibble address)
- XROM table. Actually, it is a list of all XROM "modules", each "module" has a variable number of functions.
The table is built as this:
XROM start code (3 nibbles),
number of functions in this "module" (2 nibbles),
list of the functions entry addresses (5 nibbles per address).
Then the next XROM follows with the same format. The table starts at 1be7d and ends at 1c151.
- 42S special text bytes (Fx yz, with yz>=80h)
This table is located at 1c152 and is 280h nibbles long (128 entries, one for each 'yz' value)

In each of these tables, entries can be either RPL or machine code (assembly). The 2 kinds of entry are recognized with the address MSB: if set, the entry is RPL, if not it is assembly.

3.3 The hidden XFCN keyword

The XFCN is a hidden function (it is not in CATALOG) that asks for a ALPHA argument.
The XFCN has been a mystery for me for years. I tried many times to figure what it could be used for. And I always got the same message "Nonexistent"...

So as soon as I discovered the op tables, I disassembled the XFCN code. I cannot resist to give you the result:

```
128EB 1F00002      D1=(5) #20000
128F2 143          A=DAT1 A
128F5 34F3CA5      LC(5) #5AC3F
128FC 8A690        ?C<>A A / GOYES #12908
12901 8F50002      GOSBVL #20005
12908 8C8FA3       GOLONG #16402
```

If there is a ROM at address 20000, and if the first 5-nibble word is 5ac3f, then the XFCN code jumps to 20005. Otherwise, it returns an error. Easy, isn't it? (5ac3f is also the value of the test word at RAM location 5001a)
I never opened a HP-42S, but from what I read in some posts, I think that there is a place for a second ROM. It's true for the HP17BII which have the same hardware, and it is how HP managed the two HP17BII versions (english and international with multiple languages): one common ROM and a second optionnal ROM.
So XFCN is a hook to access an optionnal second ROM, the old mystery is over now ...

3.4 The COMP bug

In some versions, the COMP function is not working properly: if arguments are not valid (for example if the arguments are in the wrong order), then the error status is latched and COMP function always returns an error even with good arguments. Error status can be easily reset by an arithmetic operation (+, -, ...).
It has been suspected that it was related to an interaction between software and hardware. I confirm this hypothesis and can give some details. I own A and C versions. On my real machines, bug occurs with version C, but not with version A. In my emulator, the bug occurs with both versions! I found that it is related to the XM hardware status flag that is used as an error flag in COMP code, but XM is not reset when entering into COMP, nor in the main loop, nor by a machine reset! (Fortunately, it is clear by arithmetic functions...). With my real A version machine, the XM flag seems to be reset at some place, but I was unable to reproduce it in my emulator.

That's all for today.

Jean-Francois Garnier (France)
jfgarnier@wanadoo.fr

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

hp-67 / hp-97 card reader repair hints (wheel & clutch)

Posted by [Bernd Schmeling](#) on 3 Apr 2002, 5:47 p.m.

I do not want to post the complete repair instructions, this is posted and documented a lot of times before. Best article of repair I found on <http://www.rskey.org/hp67reader.htm> and <http://www.rskey.org/hp97reader.htm>, in which I really easy and successful followed to repair my both hp's.

In addition to that, I like to share following experiences:

1. I was able to take the measure of the almost desintegrating drive wheel. The outer diameter of the rubber (I took the measurement of a wheel, that was nearly complete) should be 6.15 mm.
2. Clutch problem. After repair of the hp-67 hp-97 the reader worked fine for a couple of attempts. Suddenly it became poor, the mechanism did no more pull the cards smoothly and became more and more stumbling while pulling the cards. Investigations about this brought following result: The clutch between the motor and the spindle is made of a stuff, that is similar to the stuff, which the drive wheel is made of. So it did not desintegrate, like it is known by the wheel, but it is hardening, so the motor bolt is just running through. To solve this problem, I removed the spindle from the motor, pulled of the clutch and used as replacement a well fitting insulation of a wire, I found in my 'trash-box'. Now I had a really good working clutch, that was able, to transport the card smoothly through the mechanism, but if stopped by force, that released the motor to run free through this 'replacement' clutch.

Why making so much efforts to make a working clutch, instead of fixing the spindle just with 'super-glue' directly onto the motor bolt?

Because of two reasons: If there is no clutch, and if the mechanism might be blocked of any reason, what will happen is as follows: The (screw-like) spindle will keep turning to the wheel with all its power. The result will be, that the spindle will destroy the wheel.

Second what 'might' happen is, that the blocked motor could be destroyed by overheating (I don't really beleave in this, because every motor I know is able to withstand a blocking for more as a several seconds. The reader's motor will be shut down by an error message on the display of an hp after a very small time!)

So, if you like to avoid hazards like this, you must not have a fixed connection between the motor bolt and the (screw-like) spindle!

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

liffile - HP9114 LIF disk image file

Posted by [Tony Duell](#) on 20 Mar 2002, 11:55 p.m.

This article describes the files that may be found at [ftp://ftp.hpmuseum.org/hpswap/](http://ftp.hpmuseum.org/hpswap/). You may also be interested in the [LIF Utilities for linux](#).

DESCRIPTION

The HP9114 LIF disk image file format was designed as a simple way to store HP9114 disk images on a larger computer, and to allow programs to be written to process those images (including writing them back to a physical disk).

IMAGE FILE & LIF LOGICAL BLOCKS

The file consists of a raw dump of the data blocks on the disk, in increasing numerical order. There are no headers or trailers added, either for the entire image or for each block. That is to say :

Bytes 0 to 255 are LIF logical block 0
Bytes 256 to 511 are LIF logical block 1
Bytes 512 to 767 are LIF logical block 2
and so on

As an HP9114 disk contains 2464 user blocks (organised as 77 cylinders, 2 heads, and 16 sectors per track, with each sector being 1 block), an image file is always 2464*256, or 630784, bytes long.

LIF DISK PHYSICAL BLOCKS

The correspondence between the logical block numbers described in the last section and the physical sector addresses on the disk is also straightforward. The cylinders on the disk are numbered from 0 to 76. the 2 heads (sides) are called 0 and 1, and the sectors on each track are numbered from 0 to 16. Then :

Block 0 is cylinder 0, head 0, sector 1
Block 1 is cylinder 0, head 0, sector 2
Continuing on cylinder 0, head 0, until...
Block 15 is cylinder 0, head 0, sector 16
Block 16 is cylinder 0, head 1, sector 1
Continuing on cylinder 0, head 1 until...
Block 31 is cylinder 0, head 1, sector 16
Block 32 is cylinder 1, head 0, sector 1
And continuing in this manner for all 77 cylinders

PSEUDO-CODE TO WRITE AN IMAGE TO DISK

Assuming the necessary functions exist to seek the disk head to a particular cylinder (seek()) and to write a 256 byte

sector (writesector(cylinder,head,sector,data[])) then it is reasonably easy to write a program to transfer a disk image to a physical disk. The following pseudo-code gives the basic idea :

```
declare blockbuffer to be 256 bytes long
open(image file)
for cylinder=0 to 76
  seek(cylinder)
  for head = 0 to 1
    for sector = 1 to 16
      read next 256 bytes of image file into blockbuffer
      writesector(cylinder,head,sector,blockbuffer)
    next sector
  next head
next cylinder
close(image file)
end
```

REFERENCES

The logical block structure of a LIF device is given in appendix D of the HP-IL Module for the HP71 Owner's manual. The relationship between LIF logical blocks and physical disk sectors was determined experimentally.

RESTRICTIONS

Although the LIF Utilities for linux are released under the GNU public license, the only restriction placed on the use of this image file format is that no (other) restrictions may be placed on its use. Anybody is welcome to write programs using this file format (for example to convert the image files into other formats, or to transfer between physical disks and the image format) for any computer or operating system, and to distribute these programs under any license terms they choose.

AUTHOR

The LIF image file format was designed by Tony Duell, ard@p850ug1.demon.co.uk

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 49g...dedication necessary.

Posted by [Josh \(USA\)](#) on 25 Feb 2002, 1:59 p.m.

Hey all,

I've owned an hp 49g for about six months. Prior to that, I was a solid hp 48g user. I've heard of numerous bad experiences with the 49g, but hopefully this will take a second look at a calculator which certainly deserves such. I am in the engineering industry, and this machine is invaluable in my work. In symbolic operations, and speedwise, the 49g completely destroys the 48g series.

First off, I won't even try to defend the manual. It is not quite useless, but nearly so. If you want to get acquainted with the machine, you are best off reading a manual for an hp 48g+/x series calculator, since the commands largely carry over. The buttons take some getting used to, but they do break in with moderate use, and I've since gotten used to the new keyboard layout.

A few tips for new (or not new) 49g users: *Use the 49g in rpn mode. That's where it shines. *Get (or if you're adventurous, make) a pc link cable. *Upgrade to the latest BETA ROM. At the time that this is being written, that is release 19-6. Hp doesn't support it, but it fixes almost every bug I've come across, and adds a help function for a lot of commands, right in the calculator. The Beta Rom has not caused any problems in the 4+ months I've used it. *visit <http://www.hpcalc.org>. The ultimate online resource. If you want to learn to program the hp49, you need to stop in here. *PLAY, PLAY, PLAY! I learn to use it a little at a time, mostly as I need to work with new commands. This is where the manuals can be of some help. If you look up a specific command, you can generally find out some useful details on how to use it. *Talk to other users (this can be hard, since hardcore hp users are few and far between). Online forums are generally the best bet. *Take advantage of it's strengths. If you don't know how to use the equation editor, you need to learn. Use the numeric and symbolic solvers. SUBST does the same thing as ISOL, only better. *Which brings me to another point. For most mathematical operations, there are several different algorithms. I know of at least two different ways to integrate, differentiate, symbolically isolate a variable, solve a linear system of equations, etc...Look for these multiple routes to the answer. In most cases, one is more reliable.

A lot of time went into developing this machine, and I can confidently say that I have not learned half of the intricacies about it. Patience and time...

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Repairing the HP-67

Posted by [Mark Hoskins](#) on 30 Jan 2002, 1:49 p.m.

[View the Article](#) in Adobe Acrobat format (PDF)

If you don't have an Acrobat reader, readers for many platforms may be [freely downloaded](#) from Adobe.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 41C Card Reader Repair, The Short Fix Approach

Posted by [Mark Hoskins](#) on 30 Jan 2002, 1:48 p.m.

[View the Article](#) in Adobe Acrobat format (PDF)

If you don't have an Acrobat reader, readers for many platforms may be [freely downloaded](#) from Adobe.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 41 display reset.

Posted by [m robinso](#) on 30 Dec 2001, 2:02 a.m.

The display is regulated by flags 28 and 29.

Flag 28 controls the roles of periods and commas in numerical displays.

Flag 29 controls whether or not a separator mark is displayed.

The default state of both flags is set for US usage. If all else fails, try removing the batteries until the calculator is really dead and the flags should get reset.

The more elegant way to deal with this is through the set flags (SF) / clear flags (CF) commands. I have a CV and the SF is on the 7 key and CF is on the 8 key. Activate the SF key. The display will show SF __. Enter 28. The display is now back to the default.

Happy Calculating!

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Financial-Management (and Modified) Internal Rate-of-Return for the HP-17BII and HP-19BII

Posted by [W. Bruce Maguire II](#) on 29 Nov 2001, 4:49 p.m.

Here is a program to calculate Financial-Management Rate of Return (MIRR) as well as Modified Internal Rate of Return (MIRR) and plain-old IRR for comparison purposes. The program is my hacked modification to an excellent FMRR/IRR program that is listed in the Grapevine Publications Pocket Guide for the HP-19BII. Please go to: <http://www.read-gpi.com> and buy one of their excellent books! The HP-17BII has *all the same programming capability* as the 19BII, and therefore this program will work on both machines.

Obviously, the first step will be to key this program into the machine as a new formula in the "SOLVE" menu. My own convention is to name each formula, followed by a colon and two spaces for readability. This allows you to quickly find which program/formula you are interested in using. For instance, I start the following program like this:

```
FMRR%:  L(N:L(MP:L(MF:0)))...
```

You then create a cash-flow on the machine (see the User's Manual), and go to the solve menu and "CALC" this program.

For the basic problem of solving for the FMRR (or MIRR), you would enter the P/YR (number of periods per year) and the "safe" and "risk" interest rates (as Annual Percentage Rates, a.k.a. APR). Then press "\$IN" to see the "up-front" amount of capital to finance the investment, and press "\$OUT" to see the amount of capital available for withdrawal at the end of the investment. Press the "MORE" button to see the other variables in the equation: FMRR, MIRR, and IRR. Pressing the "FMR%" button will return (after a while) the FMRR.

The IRR can also be obtained (for comparison purposes), but remember that it is a less "stable" equation which may have multiple solutions! If you need to give the equation a guess for the IRR to aid in finding a solution, you can key in guess #1 and hit the "IRR" key, (optionally) key in guess #2 and hit the "IRR" key, and finally hit the "IRR" one last time to start the search for a solution (this solver functionality is explained in the manual).

In the following listings, there are three symbols that I use special notations to represent.

- 1) The multiplication symbol is "_mult_"
- 2) The division symbol is "_div_"
- 3) The summation symbol is "_sum_"

Note #1: The cash flow which is used for the calculations must be entered prior to "CALC"ing this formula, and it must be named "\$". If you really prefer another name for some reason, you can always change it in the listing. Wherever "FLOW(\$:...)" or "#T(\$:...)" or "SIZEC(\$:...)" occurs (the cash flow operations), just change the "\$" to your preferred name.

Note #2: There is a variable used for the number of periods per year: "P/YR". The "/" symbol is found in the misc. characters menu ("WXYZ", "OTHER", "MORE", "/") as are the colon and summation sign; it is *not* the divide key! Remember, the interest rates are entered as *percent per year* (not *per period*)---that is why the number of periods per

year is a required input.

Note #3: This listing is formatted using my own style for ease of understanding and reading. Spaces should not be used when typing the formula in---except in one important case. For the "OR" operation, you *must* have a space on either side of the operator, or the calculator will be confused!

Oh, one last caveat for the MIRR calculation. Since I hacked the original FMRR program to add MIRR capability, it assumes that the first cash flow is negative when calculating MIRR (this saves a lot of extra code). It seems to me that a positive initial cash-flow would be a strange (but nice!) investment.

PROGRAM LISTING:

FMRR%:

```

L(N:
  L(F:
    L(M:
      L(W:
        L(MP:
          L(MF:
            0
          )
        )
      )
    )
  )
)_mult_

(
  IF(P/YR=0:
    L(P/YR:
      1
    )
  :
    0
  )+

  L(I:
    G(IRR%)_div_P/YR
  )+

  L(S:
    SAF%_div_P/YR
  )+

  L(R:
    RSK%_div_P/YR
  )+

  _sum_(
    G:
      SIZEC($):
      1:
      -1:
      L(C:
        FLOW($:G)
      )+
      L(T:
        #T($:G)
      )+
      L(X:
        G(M)_mult_SPPV(G(S):G(T))+
        G(C)_mult_USPV(G(S):G(T))
      )+
      L(MP:
        G(MP)_mult_SPPV(G(S):G(T))
      )+

```

```
IF(G(C)<=0:
  L(MP:
    G(MP)+
    G(C)_mult_USPV(G(S):G(T))
  )+
  L(M:
    G(X)
  )
:
  L(MF:
    G(MF)+
    SPFV(G(R):G(N))_mult_
    G(C)_mult_USFV(G(R):G(T))
  )+
  IF(G(M)+G(C)>0:
    L(W:
      G(W)+
      SPFV(G(R):G(N))_mult_
      (
        G(M)+
        G(C)_mult_USFV(G(R):G(T))
      )
    )+
    L(M:
      0
    )
  :
    IF(G(X)<=0:
      L(M:
        G(X)
      )
    :
      L(P:
        IP(
          LN(
            -G(M)_mult_G(S)_div_100_div_G(C)+1
          )_div_
          LN(
            1+G(S)_div_100
          )
        )
      )+
      L(W:
        G(W)+
        SPFV(G(R):G(N)+G(P))_mult_
        (
          G(C)_mult_USFV(G(R):G(T)-G(P))+
          G(M)_mult_SPPV(G(S):G(P))+
          G(C)_mult_USPV(G(S):G(P))
        )
      )+
      L(M:
        0
      )
    )
  )
)+
L(F:
  G(F)+
  G(C)_mult_
  USFV(G(I):G(T))_mult_
  SPFV(G(I):G(N))
)+
L(N:
  G(N)+G(T)
)
)+
)

IF(S($IN):
  $IN-G(M)-FLOW($:0)
:
  IF(S($OUT) OR S(P/YR):
    $OUT-G(W)
  :
    IF(S(FMR%) OR S(SAF%) OR S(RSK%):
      (G(M)+FLOW($:0))_mult_
      SPFV(FMR%_div_P/YR:G(N))+
      G(W)+
```

```
0_mult_(SAF%+RSK%)
:
IF(S(MIR%):
  FLOW($:0)+
  G(MP)+
  G(MF)_mult_SPPV(MIR%_div_P/YR:G(N))
:
  FLOW($:0)_mult_
  SPFV(IRR%_div_P/YR:G(N))+
  G(F)
)
)
)
```

END OF PROGRAM LISTING.

Let me know if there are any bugs in my listing, or if you have any problems.

Have fun, and witness the power of these *great* calculators!

Bruce.

Edit MessageDelete MessagePassword:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

HP-12C Trigonometric Functions

Posted by **Viktor Toth** on 21 Oct 2001, 9:52 p.m.

By popular demand: a set of two programs that compute all six trigonometric functions (sin, cos, tan, arc sin, arc cos, and arc tan) on the HP-12C.

The first program at address 01 computes the cosine, tangent, and sine of the argument (in radians) and places the result in this order in X, Y, and Z. If all you need is the sine, you can stop at step 24. This program has four shortcomings:

1. For an argument of 0, the program fails, due to its use of the argument's reciprocal value in the iteration. It is possible to correct this deficiency by inserting four lines of code and adjusting line numbers accordingly:

```
01      X=0
02      INTG
03      X=0
04      GTO 29 (line 25 in the original listing)
```

2. For arguments outside the range of $-\pi/2 < x < \pi/2$, the sign of the cosine and tangent results may not be correct.
3. For arguments with a large ($> \pi$) absolute value, results are increasingly inaccurate.
4. For arguments near $n \cdot \pi/2$ (n is integer), the accuracy may be less than desirable, due to rounding effects.

The second program has three entry points (marked) for computing the arc cosine, arc tangent, and arc sine of the argument. The result is in radians. This program also has shortcomings:

1. The arc cos is calculated incorrectly for negative arguments
2. Execution time is excessive if the arc sin argument's absolute value is near 1. (A workaround is to compute the arc cos instead, and then subtract the result from $\pi/2$.)

Here are some typical results:

X	cos X	error	time	arc cos X	error	time
0.01	0.9999500004	$< 2e-11$	5 s	0.01000000167	$< 3e-10$	6 s
$\pi/4$	0.7071067810	$< 1e-10$	8 s	0.7853981635	$< 2e-10$	37 s
$\pi/3$	0.5000000003	$< 1e-9$	10 s	1.047197550	$< 1e-9$	75 s
1.77	0.1978888145	$< 6e-10$ (sign)	11 s	1.371592654	$< 3e-10$	430 s
3.14	0.9999987317	$< 3e-11$ (sign)	15 s	0.001592671386	$< 3e-7$	5 s

So, there's room for improvement.

And now, the programs themselves:

```
01      STO 0
02      STO 1
```

```
03      1/x
04      ST0 2
05      ST0+2
06      RCL 2
07      ST0/1
08      Rv
09      ST0+2
10      RCL 2
11      CHS
12      ST0/1
13      RCL 0
14      RCL 1
15      ST0+0
16      CLX
17      RCL 0
18      -
19      x=0
20      GT0 24
21      Rv
22      Rv
23      GT0 05
24      LASTx      X=sin
25      1
26      LASTx
27      ENTER
28      X
29      -
30      SQRT
31      /
32      LASTx
33      R/S      X=cos      Y=tan      Z=sin
=====
34  acos  ENTER
35      X
36      1
37      -
38      CHS
39      SQRT
40      GT0 48
41  atan  ENTER
42      ENTER
43      X
44      1
45      +
46      SQRT
47      /
48  asin  ST0 0
49      ST0 1
50      ENTER
51      X
52      1
53      ST0*1
54      ST0*1
55      1
56      +
57      ST0/1
58      1
59      +
60      ST0/1
61      x-y
62      ST0*1
63      x-y
64      RCL 0
65      RCL 1
66      ST0+0
67      CLX
68      RCL 0
69      -
70      x=0
71      GT0 74
72      Rv
73      GT0 53
74      LASTx
75      GT0 00
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Improved TRIG. and INVERSE TRIG. functions for the HP-17BII

Posted by [W. B. Maguire II](#) on 26 Sept 2001, 12:40 p.m.

Improved TRIG and INVERSE-TRIG functions for the HP-17BII

+-----+
| If _anyone_ finds this post useful, please validate me with a response! ;-) |
+-----+

Background:

OK. I *really* like the HP-17BII. It has the great Pioneer dimensions. It has a nice, big 22-character display. It has dedicated soft-keys and a second line in the display for their labels. (I think I prefer this soft-key implementation to the 42's overloading of the soft-keys. I'll be in a menu and think I have access to those functions on the top row of keys. The dedicated soft keys [pioneered by the 28c/s] seem to be more intuitive to me.)

Anyway, being an engineer, I break out in a rash if I don't have access to the trigonometric and inverse-trig. functions.

So, to borrow from "Animal House":

I think that this situation absolutely requires a really futile and stupid gesture be done on somebody's part. And I'm just the guy to do it.

Several weeks ago, I posted my routines for calculating the TRIG. and INVERSE TRIG. functions. Unfortunately, I didn't realize that the factorial function existed on the 17BII! Duh! I also did not know about the undocumented "L()" and "G()" (Let and Get) functions. I *strongly* encourage anyone who would like to program the 17BII to buy the CD-ROMs from the HP Museum in order to obtain the "Technical Applications" book for the 27S and 19B. The undocumented "Let" and "Get" functions are described, as well as methods of executing looping---these functions open-up programming options that were not conceivable before! For instance: definite loops and invisible, temporary variable assignments. Definitely a *MUST READ!* By the way, in my opinion, these functions completely put to rest the question of whether the 17BII and 19BII are "programmable"---they *absolutely* are!

So, anyway, here is my self-redemption for those embarrassing, brute-force routines.

Equation Descriptions:

The following equations for the HP-17BII calculate the TRIG. and INVERSE TRIG. functions. I originally had larger formulas that combined SIN,COS,TAN into one equation, and ASIN,ACOS,ATAN into one equation. But, I wanted to minimize the "CALC" time for these formulas, since it's just overhead. As a result, I have left out the TAN function, because it is very easy to enter the angle, press [SIN] then [COS] then the divide key. Since ASIN and ACOS are so similar, they are in one equation, but ATAN is just different enough to (in my mind) justify a separate equation.

Another approach---which I still might consider---is to cram all desired functions into one single equation, and then try not to leave the solve menu. Getting back into the solve menu would take literally minutes of "CALC" time!

In the interest of accuracy, the inverse trig. functions are divided into multiple domains, so that the polynomials converge faster. The ASIN and ACOS have two domains: $[0, \pi/4]$ and $[\pi/4, \pi/2]$. The ATAN is a little more complicated; its domains are: $[0, \sim 0.3927]$, $[\sim 0.3927, \sim 1.1781]$, and $[\sim 1.1781, \text{inf}]$. The accuracy for the ATAN function is MUCH better than a tangent Taylor series approximation (and therefore MUCH faster). After the routine listings, I have computed (using Matlab) the worst-case accuracy of the equations, given various maximum-exponent values (in the polynomial series).

In the following equations: "SIGMA" means the sigma symbol:

(In the ALPHA menu:
[WXYZ][OTHER][MORE],
then the second button)

"ANGLE" means the angle symbol:

(In the ALPHA menu:
[WXYZ][OTHER][MORE][MORE][MORE],
then the fifth button)

The formula for the SIN and COS was kindly supplied by Kynes.

Equation Listings:

DEGREES~RADIANS:
DEG/180*PI=RAD

SIN, COS, D~R:
IF(S(SIN):
SIGMA(N:0:9:1:(-1)^N*
ANGLE^(2*N+1)/FACT(2N+1)) - SIN
:
IF (S(COS):
SIGMA(N:0:9:1:(-1)^N*
ANGLE^(2*N)/FACT(2N)) - COS
:
DEG/180*PI - RAD
)
)

ASIN, ACOS, D~R:
IF(NOT (S(DEG) OR S(RAD)):
0*
IF(X<.7071:
L(R3:X)+
IF(S(ASIN):
L(R1:0)+L(R2:1)
:
L(R1=1)+L(R2=-1)
)
:
L(R3:SQRT(1-SQ(X)))+
IF(S(ASIN):
L(R1=1)+L(R2=-1)
:
L(R1:0)+L(R2:1)
)
)
+G(R1)*PI/2+G(R2)*(
G(R3)+
SIGMA(N:3:31:2:
(FACT(N-2)*(G(R3))^N/
(2^(N-2)*FACT((N-3)/2)*FACT((N-1)/2)*N)))
)
-
IF(S(ASIN):

```

      ASIN
      :
      ACOS
    )
  :
  DEG/180*PI - RAD
)

ATAN,D~R:
  IF(NOT (S(DEG) OR S(RAD))):
    0*
    IF(L(AA:ABS(SQ(X)-1)/(SQ(X)+1))<.7071:
      L(R1:.5)+
      L(R2:SGN(X-1)/2)+
      L(R3:G(AA))
      :
      L(R1:.5+SGN(X-1)/2)+
      L(R2:.5-G(R1))+
      L(R3:2*X/(SQ(X)+1))
    )
    +G(R1)*PI/2+G(R2)*(
      G(R3)+
      SIGMA(N:3:31:2:
        (FACT(N-2)*(G(R3))^N/
        (2^(N-2)*FACT((N-3)/2)*FACT((N-1)/2)*N)))
      )
    -ATAN
  :
  DEG/180*PI - RAD
)
```

Sample Calculations:

Sin(30 degrees):
"CALC" the SIN,COS equation, then press:

30[DEG][RAD][ST0][angle-key][SIN]
Answer = 0.5000

Tan(45 degrees):
"CALC" the SIN,COS equation, then press:

45[DEG][RAD][ST0][angle-key][SIN][COS][divide-key]
Answer = 1.0000

ArcSin(0.5):
"CALC" the ASIN,ACOS equation, then press:

0.5[X][ASIN]
Answer = 0.5236 radians
To get the answer in degrees, press:
[ST0][RAD][DEG]
Answer = 30.000 degrees

The ArcTan function works the same as the ArcSin and ArcCos, but you need to "CALC" the ATAN equation, rather

than the ASIN,ACOS equation.

Accuracy Calculations:

For the following accuracy calculations, Matlab was used. Therefore the following accuracies are closer to the theoretical accuracy of the previous algorithms. 'Epsilon'---in the computational context---means the smallest number (in a computer) that may be added to one, and yield a result that is greater than one. It is a measure of the precision that the computer uses to represent numbers (really the number of bits for the mantissa). The Epsilon for the HP-17BII is about 5e-12. (I presented an equation to show this for the HP-17BII in a separate posting.) The Epsilon for Matlab is about 2e-16. Therefore, the HP-17BII will not be any more accurate than 5e-12 in the best case (not O(10^-16) as indicated in some cases below).

For the SIN,COS equation (COS accuracy is the same as SIN):

Highest exponent of 5,	gives maximum normalized	SIN error of	5.62e-08
Highest exponent of 7,	gives maximum normalized	SIN error of	6.02e-12
Highest exponent of 9,	gives maximum normalized	SIN error of	5.55e-16
Highest exponent of 11,	gives maximum normalized	SIN error of	5.41e-16

Highest exponent of 5,	gives maximum normalized	SIN/COS error of	4.84e-03
Highest exponent of 7,	gives maximum normalized	SIN/COS error of	6.76e-07
Highest exponent of 9,	gives maximum normalized	SIN/COS error of	3.67e-11
Highest exponent of 11,	gives maximum normalized	SIN/COS error of	1.21e-12

For the ASIN equation (ACOS accuracy is the same as ASIN):

Highest exponent of 15,	gives maximum normalized	ASIN error of	7.10e-05
Highest exponent of 17,	gives maximum normalized	ASIN error of	3.03e-05
Highest exponent of 19,	gives maximum normalized	ASIN error of	1.32e-05
Highest exponent of 21,	gives maximum normalized	ASIN error of	5.78e-06
Highest exponent of 23,	gives maximum normalized	ASIN error of	2.57e-06
Highest exponent of 25,	gives maximum normalized	ASIN error of	1.15e-06
Highest exponent of 27,	gives maximum normalized	ASIN error of	5.19e-07
Highest exponent of 29,	gives maximum normalized	ASIN error of	2.36e-07
Highest exponent of 31,	gives maximum normalized	ASIN error of	1.08e-07
Highest exponent of 33,	gives maximum normalized	ASIN error of	4.95e-08
Highest exponent of 35,	gives maximum normalized	ASIN error of	2.29e-08
Highest exponent of 37,	gives maximum normalized	ASIN error of	1.06e-08
Highest exponent of 39,	gives maximum normalized	ASIN error of	4.92e-09
Highest exponent of 41,	gives maximum normalized	ASIN error of	2.30e-09

Accuracy for ATAN in the ATAN equation is the same as ASIN and ACOS.

Now, all the HP-17BII needs is an "Engineering" display mode!

Bruce.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-17BII precision: Epsilon

Posted by [W. B. Maguire II](#) on 24 Sept 2001, 4:33 p.m.

Calculating Epsilon on the HP-17BII

I wrote a program to compute 'Epsilon' which---in computational context---means the smallest number (in the computer) that may be added to one and yield a result that is greater than one. It is a measure of the precision that the computer uses to represent numbers (really the number of bits used for the mantissa).

I first wrote a traditional "Epsilon" equation for the HP-17BII, but I realized that the number storage was not the standard binary floating point, but rather some form of binary-coded-decimal. As a result, the original "Epsilon" program yielded an incorrect result. The general algorithm may be used on any machine that stores floating point numbers in an IEEE-like format. So, I re-wrote the equation for a binary-coded-decimal machine. The new equation yields the correct result for the HP-17BII; the Epsilon is 5.00000000001e-12 (twelve digits in the mantissa).

Binary Floating-Point Epsilon algorithm---INCORRECT for HP-17BII:

```
Epsilon = 1;
while ((1 + Epsilon) > 1) {
    Epsilon /= 2;
}
Epsilon *= 2;
```

Below is the HP-17BII equation. Please note that in the listing below "SIGMA" means the sigma symbol:
In the ALPHA menu:
[WXYZ][OTHER][MORE],
then the second button

Binary-Coded-Decimal equation for the HP-17BII:

```
EPSILON: 0*L(E:1)*
    SIGMA(N:1:25:1:(
        SIGMA(I:1:9:1:
            IF(1+L(X:(G(E)-I*10^(-N)))>1:
                L(NEW:E:G(X))
            :
            0
        )
    )
    +L(OLDE:G(E))
    +L(E:G(NEW:E))
)
+
IF(G(OLDE)>G(E):
    0
:
    G(E)
)
=EPSILON
```

The extra code concerning the "OLDE" variable is just to make sure that we have looped far enough. If Epsilon comes back as zero, then we need to search more powers of ten in the "N" loop (i.e. increasing the value of 25 here).

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Better trig functions on HP17BII using Maclaurin series

Posted by [Michael Blankenship](#) on 10 Sept 2001, 11:35 a.m.

[This is a reprint of a message I posted in the forum]

All,

I found a way to program the trigonometric functions into my HP 17BII business calculator. True story: my calculus professor told me on the first day of the course to ditch my business calculator and go get a trigonometric calc. Bummed, I leafed through the back of the calculus book and read about the Maclaurin power series. Click, click, click and now I have the trig functions programmed into my HP. You'd think that a certified calculus professor would have suggested this, huh? Anyway, I had a blast figuring out the nifty HP programming language after much struggle to find the manual, searching the Internet for a source of the programming instructions, failure and the "turning up" of the actual manual during a physical search for something else. Anyway, without further ado, the cosine function for the HP business calculator's SOLVER functionality:

SOLVE -> NEW -> COS=1+SIGMA(I:2:12:2:((MOD(I/2:2)x-2)+1)x(THETA^I)/FACT(I))

Note: Replace SIGMA above with the actual greek sigma character (looks like a capital E) from the ALPHA -> WXYZ -> OTHER -> MORE menu. Replace THETA above likewise with the greek theta character by pressing MORE twice more. The slash character above is simply the "divide by" character and the character "x" is the times sign, not the letter. The colon is also found on the ALPHA -> OTHER menu.

Since this forum doesn't provide for good formatting I can't directly show the equivalent formula but basically:

$\text{COS}(\theta) = 1 + \text{sum of series as } I \text{ steps from 2 to 12, stepping by 2 each time and alternating signs } (\theta \text{ raised to the } I \text{ power over } I\text{-factorial}).$

The modulo section is just so that the terms alternate signs, in other words:

$\text{COS}(\theta) = 1 - ((\theta^2/2!) + ((\theta^4/4!) - ((\theta^6/6!) + \dots$

Once you've typed this into the HP, press INPUT and CALC and enter in 0 and press the function key associated with theta, then press the COS function key. The answer should be 1.00000, depending upon your current number of decimal places (which should be about six for these kinds of calculations). Note that the angle input is in radians rather than degrees. More on that in a moment...

Without much further explanation (other than the fact that the SIN function uses the odd powers of I instead), here are the remaining functions you might need:

SIN=SIGMA(I:1:13:2:((MOD((I-1)/2:2)x-2)+1)x(THETA^I)/FACT(I))

TAN=SIGMA(I:1:13:2:((MOD((I-1)/2:2)x-2)+1)x(THETA^I)/FACT(I))/(1+SIGMA(I:2:12:2:((MOD(I/2:2)x-2)+1)x(THETA^I)/FACT(I)))

DEGREES:DEG=RADIANS/PIx180

Notes: The "DEGREES:" text above is just a label so that you can more easily find the conversion formula. By inputting in the cosine you want (1.00, for example) into the COS function and then pressing the function key associated with theta, you get the ARCCOS function, yielding 0.0 in this case.

Don't forget that you can store a variety of temporary numbers here and there using the [STO + number] and [RCL + number] syntax. I could have replaced the...

"((MOD(I/2:2)x-2)+1)" code with "IF(MOD(I/2:2)=1:-1:1)"

...but this resulted in more characters. I was a little disappointed to find that the IF() function doesn't evaluate expressions as C/C++ does. I could have saved a little more program space. If you want more precision, increase the number of decimal places in your calculator first and then if that's not enough, increase the upper range of the sigma function to something larger than twelve (in the cosine function, for example).

P.S. - Those of you familiar with the Maclaurin series for the SIN function might say, "Hey, the first term of the series is just theta itself." Well, yes, but I was optimizing for size of program, not speed of execution. If you feel otherwise, feel free to change the code so that the series begins at 3 instead of 1 and add theta somewhere outside of the series function itself.

P.S.S. - Okay, so why didn't I optimize out the first term of the power series for the COS function? Note that (0^0) is undefined had I started the series at the zeroeth instead of the second power.

Edited: 16 Feb 2006, 8:42 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous \]](#) [\[Next \]](#)

Increasing HP-42s Memory to 32K

Posted by [Paul Brogger](#) on 6 Sept 2001, 7:03 p.m.

Believe it or not, one may upgrade an HP-42s to 32K RAM.

Of course, I can make no guarantees about what happens if you try any of this. I have reported my experience as clearly as I can. I was able to upgrade two units a few days ago, and they seem to be working fine as I write.

I've asked for Tony Duell's comments on my methods, and those are added below. Read everything before trying anything. If you're hesitant about it, consider asking a more experienced person to do the touchy stuff.

To effect a 32K upgrade, you have to:

1. Open up the HP-42s

Pioneer calculators are held together by eight "heat stakes" – plastic poles that are molded into one half of the case, fit into holes in the other half, and are melted into mushroom, rivet-head shapes to effect a mechanical bond. The top four "rivet heads" are visible, recessed in holes beneath the battery compartment cover. These are accessible and easily undone. The lower four are arranged in a line, beneath and between the lowest two rows of keypads. The mushroomed heads are hidden behind the metal foil on the calculator face. These are inaccessible, but that turns out not to matter much.

Remove the battery compartment door and the batteries. Use a small (3/16" or so) sharp drill bit to cut out the melted tops from the four visible heat stakes that hold the top end of the calculator together. (I put drill bit in a pin vise or other holder and rotate it by hand.)

Caution: You only need to remove 1/16" of plastic or less, so **don't use a power drill**, as it will rapidly cut right on through the calculator front!

Once the top end is freed, you should be able to easily pry the front and back apart at the top end, using your fingernails in the slot that runs down both sides of the case. (Don't use a screwdriver or other prying implement – you'll mar the finish on the sides of your calculator, and risk internal damage as well.)

As the case starts coming apart, I simply keep prying with my (fairly strong) fingers. It's a bit of a struggle, but there are only the four hidden heat stakes holding the two halves together. With the right amount of force, carefully applied, they quickly pop one after the other in a zipper-like noise, and the calculator case comes apart.

Caution: Try to grip and pry the plastic case parts only. **Don't** get your fingers into the electronics, and don't bend the printed circuit board. If you apply pressure only to the case halves, the PCB is free and unstressed, and will not be damaged. **Don't** use any prying device, as (again) you risk cosmetic and internal damage to the calculator.

2. Remove the 8K RAM chip (the only 28-pin chip on the board)

I use a small, decent-quality soldering iron with a finely-pointed tip. I also have a long, pointed steel "ice pick" prying tool, but have used a large sewing needle in the past. I wear magnifying lenses (cheap reading glasses bought at a discount pharmacy) to see better, and I work under good, strong light.

I work with the soldering iron in my right hand, the pry tool in my left. I work from right-to-left along a row of pins on one side of the surface mount chip. I put the pry tool in between two pins, and while prying against the pin on the left, try to lift the pin on right. I only apply prying force, however, after I've touched the soldering iron to the right pin, and the solder is melted. (You can see the solder subtly change appearance as it heats and melts.)

It's touchy, delicate, detailed work, but easily accomplished. The final (left-most) pin on the side of the chip is lifted directly (there's nothing to pry against), and the whole chip should be free enough to lift up a bit. Then I turn the calculator around, and start working again left-to-right. The chip should come free when the last pin is lifted.

3. Solder a 62256 32K (low power, 28-pin, surface-mount) RAM in place

The new chip should be the same size as the original, and should sit in the same place, and with the same orientation as the original. (There's another set of surface-mount pads on the circuit board, but ignore them.)

I run the soldering iron over each newly-exposed pad, to level the solder there and remove any hardened "solder tails" or shorts with adjacent pins. A close, careful inspection of the pads should reveal no solder shorts (or undesired connections) to other pads and traces.

I place the new chip in position, with the correct orientation, and simply touch the soldering iron to each of the four corner pins to hold it in place. After waiting and blowing a little cool air on the chip, I start soldering the rest of the pins in place, maybe four at a time, and pausing to let things cool in between.

4. Cut one jumper pad near pin 28 of this chip

There are two "jumper pads" near the newly-installed chip. They look like a pair of upper-case letters "E", arranged back-to-back. We're going to disconnect one, and connect another.

A tiny ball of solder (maybe 1mm in diameter) should indicate the jumper that is connected. (It's between the corner of the chip and the edge of the PCB, right near pin 28.) I use solder wick (copper braid) to take this away. Alternatively, you can wipe your solder iron clean several times, and then each time touch it to the jumper blob to take some away. It'll take a few hits, but eventually it'll be removed. (Proceed slowly, and pause between each, blowing gently to cool things off.)

Interestingly, even though the solder pads seem to be designed to be connected with a tiny ball of solder, there also seems to be a connection already etched in place underneath the solder ball. This should be cut away cleanly with a razor knife.

5. Solder across the other jumper nearby

Right nearby, opposite pin 26, is another jumper pad that should be connected. I build up a blob of solder on the soldering iron, and touch it to the pads. The idea is to get the little ball adhering to both pads, to complete an electrical connection that did not previously exist.

6 Reassemble and test

Before reassembling, I shave off the tops of the four lower heat stakes, attached to the back half of the case. Then the case may be snapped back together (be sure to align the battery contact with its place in the battery well) and slip in the batteries.

Mine came up with a "Memory Clear" message. Then 2nd/CATALOG and MEM should display something like "Available Memory: 31,553 Bytes". Pretty cool!

Additional Notes:

The jumper changes alter the function of pin 26 of the chip. Originally Chip Select 2 (which is tied to Vcc), it becomes a new address line.

You want to use a low power CMOS RAM chip. Some 62256s take rather a lot of standby current and they will drain the batteries quickly. Pick one that has a low standby current.

Be sure to orient the new chip correctly. The “pin 1 end” of the chip is marked with a notch or dot, and that should be pointed to the nearby edge of the circuit board.

Of course, increasing the memory doesn't get around the problem of the lack of any form of input port . . .

Sources:

Tony Duell (ard@p850ug1.demon.co.uk) suggested the upgrade in a letter to the MoHPC Forum, and he supplied all of the important technical details.

The methods described for opening the calculator, and for removing and soldering the chips, are those that I've employed in upgrading my own two HP-42S devices.

Tony Duell's Commentary:

I offered the first draft of this document for Tony's comments. Here's what he had to say:

1. Open up the HP-42S

I [Tony] always remove the metal overlay and drill out those posts as well. If you are careful it's not hard to remove the overlay without damage. Just make sure you don't crease it when pulling it off.

2. Remove the 8K RAM chip (the only 28-pin chip on the board)

Don't even attempt it without a temperature-controlled soldering iron with a fine tip. You will damage the HP42...

Get some fine wire (5A fuse wire is ideal if you can get it). Feed it under 1 row of pins. Solder one end of it to a convenient large pad on the PCB (ground, for example).

Heat the pin nearest the other end of the wire. When the solder has melted (and not before!), pull up on the wire to lift the pin off the PCB. Then free the wire from under that pin and repeat for the other pins in the row.

Now one side of the chip is free. It's easy to bend the chip back and forth to break off the other row of pins (you don't care about the 8K RAM chip, right). Remove the body of the chip.

Now desolder the remains of the pins from the PCB one at a time.

3. Solder a 62256 32K (low power, 28-pin, surface-mount) RAM in place

I used desolder braid (solder wick, soldamop, whatever you call it) to remove the old solder. Then place the new chip in position, and solder down 2 diagonally opposite corner pins. Use the temperature-controlled iron and fine solder. I use 30swg silver-loaded solder (which is sold for surface-mount rework).

Check the chip is correctly positioned and the right way round. Check it again. Then solder the other pins down.

Don't worry too much about solder bridges when you are fitting the chip. After you've fitted it, use desolder braid to clean up any bridges.

[This, BTW, is the only way to do fine SMD rework by hand. Clean up the bridges at the end. It's how I replace chips in Voyagers, etc]

4. Cut one jumper pad near pin 28 of this chip

“Interestingly, even though the solder pads seem to be designed to be connected with a tiny ball of solder, there also seems to be a connection already etched in place underneath the solder ball.”

Yes, I found this, too . . .

5. Solder across the other jumper nearby

It's generally not hard to get solder to bridge this jumper pad.

6. Reassemble and test

Put the batteries in and test it. It'll work without the case (OK, the beeper doesn't :-)). If you don't get the right messages or the machine won't turn on, or something, take out the batteries and check your work.

I [Tony Duell] call the result a "42SV". ;-)

So, there you have it. The distilled HP-42s upgrade experience of two calculator fixers – the rookie (me) and the expert (Tony Duell).

Good luck!

-- Paul Brogger (broggerp@hotmail.com)

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-97 printer ribbon repair

Posted by [Erik Wahlin](#) on 6 Sept 2001, 1:07 a.m.

If you have a torn HP-97 printer ribbon, there might be a slim chance that it can be repaired. What I have been able to do is scrape some of the plastic ribbon coating off above each conductive strip so that wires can be soldered to them. Check to see that the thermal resistors are ok by measuring between the wide strip and each of the smaller strips after the plastic film has been scraped. Do this ofcourse on the ribbon that is attached to the thermal print head. Each resistor should measure 11.5 ohms or so. There are seven all together. If one or more of the thermal resistors measure more than this value, your print head is damaged and can not be used. Next, using a sharp pointed solder iron, apply a small amount of solder to each conductive strip. There will be 16 connections all together. It is usually a good idea not to scrape close to adjacent strips but stagger the scrapings. Do this to both sides of the torn ribbon (i.e side attached to print head, and side that will attach to the printer circuit board . Next, get some wire that is used for wire wrap. This is a nice thin gauge wire. You will notice that one of the ribbon conductors is wider than the others. This is the common conductor and will require two wires since this strip carries more current than the others. The next step is very difficult unless you have mastery over soldering small wires. Solder one wire at a time to connect the broken ribbon. Remember to use two wires for the wide conductor. When you are done, you should have something resembling the picture attached to this post. Next, use an ohm meter to measure the thermal resistors to make sure you don't have any shorts or open connections.

The final step is to use some 5 minute epoxy and smear a layer over the wires and where they are soldered so that it will prevent the wires from coming loose, and hold all the wires together. One final point. To remove the printer ribbon from the printer circuit board connector, find a piece of .005" thick or so piece of metal shim. Cut it to so that it is as wide as the ribbon. Insert it into the ribbon connector directly above the ribbon. This will bend back the connector tabs so that the ribbon + shim can be removed by gently pulling them both straight out from the connector.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Interfacing IL2000 System to Analogue World

Posted by [Klug Christoph](#) on 13 Aug 2001, 2:17 p.m.

Despite it being the year 2001, activities with the IL2000 system goes on, connecting HP-41 handheld computer and the HP-IL Interface-Loop to the future. The new ANA2000 module expands the IL2000 hardware board family, adapting its nice interface system to analogue world ! ANA2000 is a multi channel converter module which includes 8 analog input lines plus 8 analog output lines. For all analog I/O channels the voltage range is plus/minus 10V with 12 Bit resolution.

For ADC I use the proven 12 Bit analog digital converter design from I/O-Board, the ADS7800 from Burr&Brown (with internal sample & hold, reference voltage and oscillator). For ANA2000 board I added an 8-channel multiplexer front-end by using two AD7511 from Analog Devices (with latch-proof input lines).

Also for DAC, I take chips from Burr&Brown - two quad converter DAC4813 (with internal data-latches and reference voltage). Setting output voltages is possible independently for any channel or simultaneously for all channels by transferring individual bytes or the complete byte stream pattern by HP-41 control software.

Addressing the input multiplexer and the DAC's is done by secondary addressing, this was used for the first time by the IL-Rack (I/O-Board manual IIX.21) and later by IL-Messlab (I/O-Board manual page IIX.25). Primary addressing is done by the well known control functions OUT1, OUT2. These commands select one hardware board like DIG2000 (16 bit digital input/output), I/O2000 (adapter board for using plug in module family from I/O-Board) and now the new ANA2000 board. The needed hardware for primary addressing is placed on the IL2000 interface board.

```

      16-bit output data word
A A A A . X X X X . X X X X . X X X X
4 address bits      12-bit data word
    
```

For realizing secondary addressing I split the transferred 16-bit output word in a 12-bit data word plus a 4-bit address word (the highest significant nibble). Using these 4-bits makes possible the ability to sub-addressing four, 12-bit data latches directly and, with additional address decoder hardware, up to 16 different address lines. For the ANA2000 module, two 3 to 8 line decoders, 74HC138, set the needed address lines for the multiplexer and the DAC's.

The most significant 4 bits of each 16-bit input data word can be used as an optional 4-bit digital input word. In this case, you need additional software which splits digital input word from the 12-bit ADC analogue data word. Normally, the highest 4-bits are not used, so jumpers set the input lines to GND.

The complete ANA2000 board has a max power consumption of nearly 3 Watts, delivered by a low drop voltage regulator and the DC/DC-Converter from the POW2000 board. If you need two ANA2000 boards for a larger application (i.e. 16 analogue I/O channels) then mount the optional DC/DC-Converter onto the POW2000 board and wire belonging power supply lines to the second ANA2000 board !

If you are HP-41 enthusiast and like to actively work with HP-IL, then I recommend the nice IL2000 system as a proven, battery-powered hardware platform. In short time you can realize your individual interface solutions, such as measurement and control applications to and from the digital and analogue world. For more detailed information now

read the expanded IL2000 manual. To get this, the I/O-Board manual, parts or finished and tested interface units than please contact the author :

Christoph Klug, Körnerstrasse 47 B, 31141 Hildesheim, Germany email : christoph.klug@de.bosch.com

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

TRIG and inverse TRIG functions on the HP-17bII

Posted by [W. Bruce Maguire II](#) on 31 July 2001, 7:13 p.m.

The following equations were written to approximate the trig. and inverse trig. functions on an HP-17bII. The error in the trig functions is on the order of 10^{-8} , and the error of the ARCTAN function (and the ARCCOS and ARCSIN) reaches a maximum at $X=0.494$ and $X=0.869$; the error at those locations is just less than 2×10^{-8} . ALL FUNCTIONS ARE ONLY FOR ANGLE BETWEEN 0 AND $\pi/2$ (i.e. 0 and 90 degrees)!

I used the "angle" symbol to represent the angle in the SIN and COS functions. In the listings below I substituted "{angle}" for this non-ascii character. This symbol can be found in the miscellaneous character menu when typing.

To calculate SIN, COS, or TAN, choose the SIN,COS,... equation from the list. Verification will take about 15 (!) seconds. Enter the angle in RADIANS, and solve for ARCSIN or ARCCOS. To solve for TAN, simply solve for SIN, then solve for COS, then hit "/".

To calculate ARCSIN, choose the ARCSIN... equation from the list. Verification will take a while (30- 40seconds or so). Enter X, and solve for the ARCSIN in RADIANS.

To calculate ARCCOS, solve for ARCSIN of the value FIRST, THEN select the ARCCOS equation, and solve for ARCCOS (it's just $\pi/2$ - ARCSIN).

To calculate ARCTAN, choose the ARCTAN... equation from the list. Verification will take about 45 (!!) seconds. Enter X, and solve for ARCTAN in RADIANS.

The following are copied from a printer listing of my equations. HOPEFULLY, I didn't introduce any typos.

I STRONGLY ENCOURAGE YOU TO VERIFY THESE EQUATIONS AFTER TYPING THEM IN! To verify SIN and COS, verify that you get $1/\sqrt{2}$ for the SIN and COS of $\pi/4$. To verify ARCSIN, you must check all three regions of calculation: X in $[0, 0.494)$, X in $(0.494, 0.869)$, and X in $(0.869, \text{infinity})$. Verify ARCTAN the same way you did ARCSIN.

If you would like this information in a text file (for easier reading), please contact me at: Maguire@AnalyticInvestments.com. Thanks!

ALL FUNCTIONS ARE ONLY FOR ANGLE BETWEEN 0 AND $\pi/2$ (i.e. 0 and 90 degrees)!

Have fun typing! And wish---like I did every second ---that you had cut-and-paste!

DEG~RAD: $\text{DEG}/180 \times \pi = \text{RAD}$

SIN,COS,[0, $\pi/2$]: IF(S(SIN):
{angle}-
{angle}^{3/6}+

$\{\text{angle}\}^{5/120}$ -
 $\{\text{angle}\}^{7/5040}$ +
 $\{\text{angle}\}^{9/362880}$ -
 $\{\text{angle}\}^{11/39916800}$ +
 $\{\text{angle}\}^{13/6227020800}$ -
 $\{\text{angle}\}^{15/1307674368000}$ +
 $\{\text{angle}\}^{17/355687428096000}$ -SIN:
 $\text{PI}/2 - \{\text{angle}\}$ -
 $\text{PI}/2 - \{\text{angle}\}^{3/6}$ +
 $\text{PI}/2 - \{\text{angle}\}^{5/120}$ -
 $\text{PI}/2 - \{\text{angle}\}^{7/5040}$ +
 $\text{PI}/2 - \{\text{angle}\}^{9/362880}$ -
 $\text{PI}/2 - \{\text{angle}\}^{11/39916800}$ +
 $\text{PI}/2 - \{\text{angle}\}^{13/6227020800}$ -
 $\text{PI}/2 - \{\text{angle}\}^{15/1307674368000}$ +
 $\text{PI}/2 - \{\text{angle}\}^{17/355687428096000}$ -COS)

ARCSIN,[0,PI/2]: IF(X<0.494:
 X+
 $1.66666666667E-1 \cdot X^3$ +
 $7.5E-2 \cdot X^5$ +
 $4.46428571429E-2 \cdot X^7$ +
 $3.03819444444E-2 \cdot X^9$ +
 $2.23721590909E-2 \cdot X^{11}$ +
 $1.73527644231E-2 \cdot X^{13}$ +
 $1.396484375E-2 \cdot X^{15}$ +
 $1.15518008961E-2 \cdot X^{17}$ -ASIN:
 IF(X<0.869:
 $\text{PI}/4 + 0.5 \cdot (2 \cdot X \cdot X - 1)$ +
 $1.66666666667E-1 \cdot (2 \cdot X \cdot X - 1)^3$ +
 $7.5E-2 \cdot (2 \cdot X \cdot X - 1)^5$ +
 $4.46428571429E-2 \cdot (2 \cdot X \cdot X - 1)^7$ +
 $3.03819444444E-2 \cdot (2 \cdot X \cdot X - 1)^9$ +
 $2.23721590909E-2 \cdot (2 \cdot X \cdot X - 1)^{11}$ +
 $1.73527644231E-2 \cdot (2 \cdot X \cdot X - 1)^{13}$ +
 $1.396484375E-2 \cdot (2 \cdot X \cdot X - 1)^{15}$ +
 $1.15518008961E-2 \cdot (2 \cdot X \cdot X - 1)^{17}$ -ASIN:
 $\text{PI}/2 - ((1 - X \cdot X)^{0.5})$ +
 $1.66666666667E-1 \cdot (1 - X \cdot X)^{1.5}$ +
 $7.5E-2 \cdot (1 - X \cdot X)^{2.5}$ +
 $4.46428571429E-2 \cdot (1 - X \cdot X)^{3.5}$ +
 $3.03819444444E-2 \cdot (1 - X \cdot X)^{4.5}$ +
 $2.23721590909E-2 \cdot (1 - X \cdot X)^{5.5}$ +
 $1.73527644231E-2 \cdot (1 - X \cdot X)^{6.5}$ +
 $1.396484375E-2 \cdot (1 - X \cdot X)^{7.5}$ +
 $1.15518008961E-2 \cdot (1 - X \cdot X)^{8.5}$ -ASIN))

ARCCOS: ACOS=PI/2-ASIN

ARCTAN,[0,INF]: IF(ABS(X*X-1)/(X*X+1)<0.494:
 $\text{PI}/4 + \text{SGN}(X-1) \cdot 0.5 \cdot (\text{ABS}(X \cdot X - 1)/(X \cdot X + 1))$ +
 $1.66666666667E-1 \cdot (\text{ABS}(X \cdot X - 1)/(X \cdot X + 1))^3$ +
 $7.5E-2 \cdot (\text{ABS}(X \cdot X - 1)/(X \cdot X + 1))^5$ +
 $4.46428571429E-2 \cdot (\text{ABS}(X \cdot X - 1)/(X \cdot X + 1))^7$ +

3.03819444444E-2*(ABS(X*X-1)/(X*X+1))^9+
2.23721590909E-2*(ABS(X*X-1)/(X*X+1))^11+
1.73527644231E-2*(ABS(X*X-1)/(X*X+1))^13+
1.396484375E-2*(ABS(X*X-1)/(X*X+1))^15+
1.15518008961E-2*(ABS(X*X-1)/(X*X+1))^17)-ATAN:
IF(ABS(X*X-1)/(X*X+1)<0.869:
PI/4+SGN(X-1)*0.5*(PI/4+0.5*(1-8*X*X/SQ(X*X+1))+
1.66666666667E-1*(1-8*X*X/SQ(X*X+1))^3+
7.5E-2*(1-8*X*X/SQ(X*X+1))^5+
4.46428571429E-2*(1-8*X*X/SQ(X*X+1))^7+
3.03819444444E-2*(1-8*X*X/SQ(X*X+1))^9+
2.23721590909E-2*(1-8*X*X/SQ(X*X+1))^11+
1.73527644231E-2*(1-8*X*X/SQ(X*X+1))^13+
1.396484375E-2*(1-8*X*X/SQ(X*X+1))^15+
1.15518008961E-2*(1-8*X*X/SQ(X*X+1))^17))-ATAN:
PI/4+SGN(X-1)*0.5*(PI/2-((4*X*X/SQ(X*X+1))^0.5+
1.66666666667E-1*(4*X*X/SQ(X*X+1))^1.5+
7.5E-2*(4*X*X/SQ(X*X+1))^2.5+
4.46428571429E-2*(4*X*X/SQ(X*X+1))^3.5+
3.03819444444E-2*(4*X*X/SQ(X*X+1))^4.5+
2.23721590909E-2*(4*X*X/SQ(X*X+1))^5.5+
1.73527644231E-2*(4*X*X/SQ(X*X+1))^6.5+
1.396484375E-2*(4*X*X/SQ(X*X+1))^7.5+
1.15518008961E-2*(4*X*X/SQ(X*X+1))^8.5-ATAN))

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

UPDATE: New batteries in old HPs

Posted by [W. Bruce Maguire II](#) on 31 July 2001, 3:34 p.m.

Hello all:

I just wanted to amend the previous post I had written about modifying a charger to charge 1600 mAh NiMH batteries. It appears that I recommended too long a charge time. It seems that about 4 hours is more appropriate for the charge current of 175 mA that I quoted.

I figured this out by looking at the charging current of two "smart" NiMH chargers I have. The energizer (I think) charger used 135 mA to charge AAs in about 6 to 12 hours, then dropping to 25 mA as a maintenance charge. The quick-charge charger I have used 225 mA to charge in 90 minutes, then dropping to 17 mA for a maintenance charge.

Please base your charging times on theses numbers. My recommendation would have resulted in charging the batteries for too long.

Thanks, Bruce.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

New batteries in old HPs

Posted by [W. Bruce Maguire II](#) on 27 July 2001, 4:24 p.m.

Hello all:

I just wanted to share my recent battery adventures with the faithful. I made a few battery packs for the HP classics using new NiMH 1600 mAh batteries (Sanyo industrial and Kodak retail). Unfortunately, a standard classic charger will take 36-48 hours to fully charge these batteries (and they might not fully charge since the current is relatively low). So, I modified a charger to output about 180mA on the battery-charger circuit rather than the normal 50-60mA. This should yield about a 12 hour charge time. It was a very easy modification. In my wall transformer, there is a 13 Ohm resistor being fed into a transistor. This resistor determines the current available from the "current source" circuit on the battery-charging side. By putting two 10 Ohm resistors in parallel with the 13 Ohm, I was able to get the net resistance to about 3.7 Ohms. This resistance theoretically should yield 190mA, but I measured less--about 175mA.

Now, the really good news: these 1600mAh NiMH batteries last a LONG time. I decided a good test would be to start my HP-55 timer (with hundredths-of-a-second displayed---meaing eight digits and two colons), and see how long it lasted. I started to get the all-decimals low-battery indicator after 9 HOURS and 45 MINUTES. I finally gave up (and went to bed) after 10 HOURS and 20 MINUTES (it was still running fine at that point).

A test in my HP-67 yielded similar results, in that a running program lasted about 8 HOURS. Unfortunately, the 67's low battery LED comes on after only an hour or two---even though the calculator runs many more hours. I haven't quite figured this one out. I'm not sure why the low-battery threshold on the 67 would be different than the 55.

Anyway, I'm a happy camper! 8 to 10 HOURS on a charge. Woo-Hoo!

Bruce.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

HP-97 Card Reader Repair

Posted by [Mike Davis](#) on 17 July 2001, 6:03 p.m.

Repairing the HP-97 Card Reader

The photos below may be of use to those of you who wish to repair the gummy wheel on HP-97 card readers. The focus of this page is to show **how** and **what** to replace. It is intentional that it does not provide suggestions for dealing with the clutch. I do not have a solution for this, that I want to commit to, at this time. It is also not a tutorial on how to treat electronic devices. You should consider your 97 as static sensitive and treat it accordingly.



If you have signs of gummy substances on your cards, after running them through the reader, the drive wheel may need to be replaced. Other signs are irratic card movement, no movement, slow movement or errors.



The first thing you must do is to remove the case back. This is not difficult but there are some precautions you must be aware of. Just remove the 6 screws shown and the case back can be removed. You may want to remove the battery and battery cover first.



To remove the interior support frame, remove the screws shown. The card reader is mounted under this frame, in the upper right.



After these 7 screws are removed, you can flip the main frame over to the left but first you should understand how the main PCB connects to the keyboard. Failure to understand this can lead to bent pins when reassembling.



Simply lift the main PCB and it will come free of the keyboard/display connector. When you reassemble this, be careful to observe proper mating, or you may wind up with bent pins.



You will notice that there is a slot in the frame. This slot is to allow the keyboard/display contacts to pass through the frame. When assembling, be careful to see that these contacts pass through this slot. If you don't... you get it... more bent pins.



Now that the frame is free of the case and keyboard, simply flip it out to the left. Be careful not to put too much tension on the printer switch connections. These can break off easily. You may want to make note of the colors and positions of wires. Leave a short strip of paper in the reader. I have found this makes testing easier.



The card reader is mounted to the frame by 3 silver colored screws. There is also a ribbon cable that connects to the PCB as shown in the photo. Two of the screws are beneath the ribbon cable.



Carefully pull the ribbon straight out to the right, to disconnect it from the reader. Use your hand **and NOT** a screwdriver that is shown in the photo. That is simply there to hold the ribbon back for the photo. Then remove the three silver colored screws. This will free the card reader from the frame.



You may want to make note of the colors of all wires and their connection points. These wires tend to break off when opening up the reader. Remove the 3 silver colored screws shown in the photo.



Flip the card reader over and take a look at the drive wheel. You may see goo on the wheel. You may want to perform this step before removing screws but it can be done in any order. This photo does not show the goo that is typical but will be replaced anyway because of age.



After you remove the 3 silver colored screws, you can move the PCB to the side of the reader. A couple of cautions here. Remove the tape to free the red and black wires, if they are taped to motor. Also, be careful to avoid damaging the switch contacts. Then, remove the two screws shown, in the photo.



This part is tricky. Flip the card reader over and grasp in the left hand as shown. Then remove the last screw at the back of the motor. With the right hand, grasp the motor half of the reader and pull it off of the switch half. You must observe this orientation (**feet up**), or you will lose then nylon balls.



If you have done this correctly, you should see 4 nylon balls and one pinch roller on the left half and two copper spring clips on the right half. Remove these for safekeeping, until they need to be installed. You should clean these parts before reinstalling. Remove the slotted shaft, indicated in the photo, to remove the gummy drive wheel.



Gummy wheels typically look like this. Clean all the gum off of the drive wheel. Be sure to check and clean the interior of the wheel where the shaft is inserted. This can sometimes have goo and will slow the motor.



I use pink silicone tubing that I get from a local model shop. The outside diameter is almost perfect and the inside diameter is tight enough that I don't have to use any glue. Push the tubing over the shaft and make sure it fits evenly all the way around.

Trim excess tubing flush to the metal. Make sure the tubing does not extend beyond the metal end. If it does, it may drag on the frame. I trim it a little smaller to avoid rubbing.



Clean the card reader surfaces and the read/write head before putting the reader back together. Install the nylon balls, pinch roller and spring clips. Make sure they are clean before installing.



Reverse the process that you used to disassemble. The left hand grasps the half that has the nylon balls (feet up) and the right hand places the other half on top. Replace the screw shown in the photo. **Tip: Practice this without the loose parts first.** Finally, install the reader PCB and replace the 3 silver color screws.



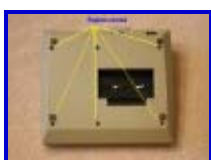
To test the card reader, assemble the frame and case as shown. Don't forget to connect the keyboard and display connector. Then, pull the ribbon cable through the frame and attach the card reader. This makes it easy to test without assembly and disassembly. Adjust slotted shaft until the card reads and writes properly.



When you have the card reader working properly, remove the ribbon cable. Thread it back through the slot and reinstall the card reader. Reconnect the ribbon cable. Check for broken wires before closing case. Note: There are slots in the reader feet. Adjust position of reader so that cards do not touch case, as card exits case slot. This takes practice.



Install the 7 screws that hold the frame in place. Check that the keyboard/display connection is mated properly. Check that you have not disconnected the printer. NOTE: Be careful with the top center screw. The screw post is fragile and breaks easy. Just make it snug.



Replace the final 6 screws in the case and your done. That wasn't hard, was it?

Remember above when I said "don't touch these contacts"? Well, I have found on many units (don't fully understand this) that I can correct "write errors" by simply wiping [these contacts](#) with steel wool. Be very careful to avoid bending these contacts. They are very hard to adjust.

I'm sure there are many other ways of making these repairs. This is just the method that works well for me.

Copyright (2001) Mike Davis

Send comments, suggestions or corrections to [Mike Davis](#)

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-67 Card Reader Repair

Posted by [Mike Davis](#) on 17 July 2001, 6:02 p.m.

Repairing the HP-67 Card Reader

The photos below may be of use to those of you who wish to repair the gummy wheel on HP-67 card readers. The focus of this page is to show **how** and **what** to replace. It is intentional that it does not provide suggestions for dealing with the label or the clutch. I do not have a solution for these two areas that I want to commit to, at this time. It is also not a tutorial on how to treat electronic devices. You should consider your 67 as static sensitive and treat it accordingly.



If you have signs of gummy substances on your cards, after running them through the reader, the drive wheel may need to be replaced. Other signs are erratic card movement, no movement, slow movement or errors.



The first thing you must do is to remove the case back. This is not difficult but there are some precautions you must be aware of. 1) Do not pry, from the sides or top, to remove the top 2 screw hole plugs. 2) Be careful in lifting the label corners. 3) The feet, at the bottom, do not come off. Just lift outside edges.



Here is what the gummy wheel typically looks like.



Remove the 6 screws shown. You can lift the motor drive circuit board where it connects to the main circuit board. This allows you to remove the main PCB and easier access to the upper right screw, which is under the circuit board.



After you remove these 6 screws, the interior electronics and boards can simply flip out to the right as shown. Be careful, the power switch and program switch contacts sometimes stick to the keyboard PCB. Remove these and set aside so they don't get lost.



Slide the top half (part with keys) off to the side until reassembly is needed. You may decide that you wish to clean keys, etc. Make sure you know where they go before you remove them (get a photo). The keyboard matrix can now simply be unfolded to the left as shown.

Remove **only** the 5 screws shown in the photo. DO NOT touch the silver colored screws that



secure the switch contacts to the board.



This is the last screw holding the card reader together. After you remove this screw, be careful not to pick things up. If you do, you may lose the small nylon balls that operate the switches. Almost everyone has lost one, at least once. **You've been warned.** You may also remove the gold contacts in the charger connector.



Now, simply lift card reader half, with the card head, and flip it to the left. You should locate the 4 nylon balls, the pinch roller and the two leaf springs. These have legs of their own and want to hide from you. Find them and secure them before you go any further



It should be obvious what needs to be removed. Just look for goo. Remove the drive wheel by pulling out the slotted drive shaft, indicated in the photo. You can't actually see it in the photo. The yellow line points to it's location. Once you get this far, you'll easily be able to see it.



Now you can see why your card reader may not pull card through the reader. Carefully clean off all the goo. You may also want to check and see if any got into the hole in the center of the wheel. This can cause slow motor if you forget to clean it out.



I use pink silicone fuel tubing that I get at a local model shop. It's outside diameter is just about perfect and the inside diameter is slightly smaller than the outside diameter of the drive shaft. When you buy this material, check for concentricity of the hole to the outside diameter. If it's not concentric, you may get a variable speed when through.



Trim the silicone tubing to fit flush to the edge of the drive wheel. You don't want to leave tubing extending beyond the shaft hole or it might rub against the side of the card reader housing and cause slow motor. I don't use any glue. The fit is tight because of the size of the tubing.



Carefully clean all the interior parts of goo or other residue. Don't forget to clean the pinch roller slot. Sometimes goo get inside the two small depressions where the pinch roller sits. Install all 4 nylon balls, the pinch roller and the 2 leaf springs. You can also install the gold charger contacts, at this time.



Make sure that none of the wires are in the area that the card reader goes. Now, carefully, flip the card reader back into position. It takes a small pressure to the left. If you do this right, it will drop right back into place.



Before you lift anything off the table, make sure you put this first screw back into it's hole. This screw is the one screw that allows you to pick this up. Failure to do so, may result in those little balls taking wing. **You've been warned again.**

Now that the initial card reader screw is in place ([see previous step](#)), you may flip the keyboard half open and reinstall the 5 screws that hold the card reader together. Make sure that these 5 screws are all the way in. Sizes of screws vary and if the heads extend above the



board, they will short out on the keyboard circuits.



Install the main PCB. Plug the motor drive board, into the main PCB. Install the "power" and "pgrm" switches and contacts. I apply a fresh coat of silicone grease to contacts. Then, carefully, place the keyboard and card reader into the calculator case. Secure it into place with the six screws.

Now that you have everything back into the calculator, you can perform a test before attaching the case back. It's relatively easy to hold the battery in place while running a card through the reader. If everything works, you can install the case back. If the cards don't read properly, you can try adjusting the shaft that goes through the reader. I usually start by turning the slot until the card does not easily push into position. I of course do this with no power initially.

Copyright (2001) Mike Davis

Send comments, suggestions or corrections to [Mike Davis](#)

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)
[\[Previous | Next \]](#)

Golden Ratio and HP-1xC series

Posted by [John Ioannidis](#) on 4 July 2001, 12:22 p.m.

Did you know that the HP-10C/11C/12C/15C/16C calculators are in the shape of a golden rectangle? No wonder they look so good!

Edit Message

Delete Message

Password:

[\[Return to the Message Index \]](#)

Go back to the main exhibit hall

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 9810/9820 Card Reader Drive Wheel Repair

Posted by [Katie Wasserman](#) on 6 Apr 2001, 4:29 p.m.

Like the card reader drive wheel on the 65/67/97/41, the drive wheels on the 9810 and 9820 card readers deteriorate and turn to goo over time. I've found a fast and easy replacement for them -- here's how:

- 1) After opening the machine (see <http://www.hpmuseum.org/disasm.htm>) remove the card reader circuit board. Use the red lever at the back of the card to start with then carefully lift it straight up. The card reader is connected to the power supply housing with a keyhole type binding post attachment so you need to wiggle it a little to free it completely.
- 2) Remove the edge connector (with the wire harness going to the card reader assembly) on the front of the card.
- 3) Remove the small daughter circuit board from the card reader circuit board going to the read/write head (it just pulls straight off) and set the card aside.
- 4) Remove the 4 screws at the base of the card reader assembly. The assembly should now be completely free with two wire harnesses hanging off it.
- 5) Put the assembly on your work surface with the drive belt face down.
- 6) Remove the 2 screws holding the read/write head in place, remove the r/w head and set it aside. This step isn't crucial, but it makes it easier to work on the assembly and keeps the head from any potential damage.
- 7) Remove the remaining 5 (or possibly 6) screws on that same side of the assembly the head was attached to.
- 8) You can now pull the assembly apart, but before doing so, you should take note of where the various parts go, in case you pull out more parts than you need to.
- 9) Slowly pull the assembly apart, as you do you'll notice (depending on how yours is wired up) that some of the photo sensors and/or light sources are wired together and this will keep you from being able to get the assembly opened up completely. The sensors and sources can be easily removed by just pulling them out of place. They are not glued, screwed or other wise attached, so you can carefully pull them out. With luck, you'll only need to pull out one or two of them. You can desolder them if you need to, but you should not have to.
- 10) You only need to disassemble the reader to the point of being able to get to the end of the shaft with the drive wheels on it.
- 11) Now, clean the goo off the drive wheels really well. You'll probably need to scrape it off with a razor blade or sharp flat screwdriver blade.
- 12) Next take three 5/8" ID x 7/8" OD x 1/8" thick O-rings.
- 13) Put one around each drive wheel and one in between the wheels. This third o-ring serves to keep the other two

properly spaced and sits low enough that it won't touch the card as it passes through. The o-rings fit very tightly and you'll need a needle nose pliers to pull them on without them twisting. You should find that the rings are so tight that there is no need to glue them down.

14) Make sure that all three O-rings touch each other all the way around. They should look a little like the Michelin Man!

15) You're done. Just reverse all of the above steps to reassemble the reader and re-install it in the calculator.

While you're at it, you might want to replace the drive belt on the other side of the assembly. Use another O-ring for this, size: 2-1/2" ID x 2-3/4" OD x 1/8" thick.

The o-rings are available from Small Parts Inc. (www.smallparts.com):

5/8" x 7/8" x 1/8" part #: U-ORB-208 10 for \$1.05

2-1/2" x 2-3/4" x 1/8" part #: U-ORB-230 10 for \$4.80

Edited: 2 May 2014, 11:43 a.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Previous | Next \]](#)

Reseting an HP41

Posted by [Steve \(Australia\)](#) on 24 Mar 2001, 6:47 a.m.

Quite often people post questions in the forum asking how they can reset their HP41 with or without losing the programs.

It seems that HP41's do tend to lock up more than we'd like to think they do, especially when turning them on the first time for a while, or when connecting some odd peripheral.

Perhaps it's static, perhaps it's the ghost of long forgotten TI calculators extracting revenge? I don't know.

Anyway, here is a list of things you can try to reset the calculator. Note that these are not my ideas they come from an excellent book "Extend Your HP41" by W Mier-Jedrzejowicz. I recommend you get a copy!

Following is a slightly updated copy of something I posted in the forum some time ago, and that I refer people to. However it's probably easier to find here.

In the book Extend your HP41, section 4.1, a number of methods are given to revive a calculator that won't turn on.

- 1) Press backarrow and/or shift to see if the blank display is caused by alpha blanks in the X register. *To see how deceiving this is, turn on your HP41, press Alpha, then Shift STO . X, then alpha. Turning the HP41 on and off will appear to do nothing. Quite evil!*
- 2) Press on then backarrow, then on, then backarrow a few times, pausing about 10 seconds between keystrokes
- 3) Remove all modules and try again (there's also mention of the use of an AM radio here too, as well as chacking for dirty battery terminals, flat batteries, and incorrectly inserted batteries). *When the HP41 is active it radiates some RF and an AM radio (remember those?) not tuned to a station can often pick these up. The sound will change when the calculaor is doing different things, and thus you can tell (maybe) if the calculator is turning on, even in the absence of a display)*
- 4) IF YOU HAVE A NEWER HP41, try pressing backarrow and enter then on, then releasing all three. *The newer HP41's have a keystroke that does sort of a soft-reset invoked by this combination of keys. BEWARE this will cause a hard reset (losing memory) if tried on an older HP41*
- 5) If you have a wand try reading some barcode. *Reading barcode will turn on the calculator. It will also tell you if the batteries are supplying power by checking for the red light!*
- 6) Try pressing R/S
- 7) Attach a card reader and run a card through it. *This will also turn on the calculator, but be careful of the card you choose because it will be read. Actually on this point using the first card of a WALL set (if it reads) will allow you to*

hard reset the calculator by interrupting it. If your batteries are flat, the card may stop part way through -- change the batteries and reinsert the card reader.

8) Take out battery and reinsert. then press ON a few times. *This also seems to reset the calculator, but this may have been the reason why you lost control in the first place! This will also clear the date and time if you have a CX or a Time Module.*

9) *Obscure Crashes due to errors in the buffer error (typically after synthetic programming mistakes) have equally obscure methods to recover from them.*

10) memory lost (aaagh!) *As per the manual, hold the backarrow, press and release the ON button then release the backarrow. This will alledgedly work if the calculator can be turned off, so strange crashes may be due to the calculator not being able to be turned off Hmmmmm.*

11) take out batteries for a LONG time OR insert batteries the wrong way (using old batteries) (POOR ADVICE IN MY BOOKS), *This discharges the internal capacitor that maintained the HP41 battery. The HP41 is protected by reverse polarity by diodes, so this shouldn't damage the HP41 OR short the battery terminals (place wire or foil over the gold connectors. It's the outside two that you need to short, but you can connect all of them without any problems). I prefer this because there's no chance of killing the calculator. Either way -- 15 minutes is advised. I actually used the reverse battry approach on a CV in the early 80's and it worked (I only left the catteries in that way for a minute or two though)*

12) Really nasty. *The sugestion is that the battery may be dead or dying, or perhaps there's a bad connection. But you would have tried new batteries first, right? Apparantly in older HP41's low batteries can cause the display drivers to fail to synchronize. A recommendation is to take the calculator apart and leave it that way for a while! -- I told you it was nasty.*

13) Even worse. *Send to HP to get it repaired.*

I used to have a card with an autoexecute program on it. The program was simply STO c. That worked for me once too. It is actually mentioned in step 10 in this book too.

Good luck.

Also mentioned (in step 12) is to put the calculator on a hard surface, lift up one end about an inch, and set it down fairly firmly. I hav also heard "drop one end about an inch" as an alternative. I'm not advocating this either.

I've found that it may also be worth squeezing or gently twisting the calculator. This is especially true if there are some intermittent faults. If this makes any difference, then it points to bad internal connections, a problem which is reasonably common.

If the calculator rattles and seems to hace some loose plastic inside of it, especially if it looks like it's been dropped or opened, then suspect the internal posts as being broken. This will eventually cause problems as the tension proveded by these posts holds internal contacts together

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

Batt. Pack for HP 80 & HP 67

Posted by [Thomas G. Magee](#) on 9 Mar 2001, 4:17 p.m.

I found that Battery Warehouse, 434 S. Market ST, Frederick MD 21701 (301) 662-2288 replaced the cells for me in a few days. \$25 ea. including tax.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Impressions on the HP-49G

Posted by [Ernie Malaga](#) on 2 Mar 2001, 9:12 a.m.

IMPRESSIONS ON THE HP-49G

by Ernie A. Malaga

The HP-49G is an incredible machine. One has only to compare it to the HP-65 to realize how much Hewlett-Packard's programmable calculators have advanced in a quarter of a century. Still, the newer machine is in some respects inferior to, say, an HP-41CX. Rather than simply join those who sing the praises of the 49G, I'd like to play the role of Scrooge and grumble and humbug and rant about what I don't like.

Complexity

For me, the sheer complexity of the 49G is quite intimidating. Gone are the days in which you could feel comfortable about the calculator, knowing that it held no further surprises -- that you knew it. With 688 different commands (in the base 49G, with no plug-ins), and no help text anywhere, it becomes a daunting task to figure things out.

More isn't always merrier.

Take this as an example. I wanted to know how many commands were available. I knew that the command was CAT, but I did not expect to find it in an unshifted key. So, after losing a minute looking for it in blue and then in red, I gave up and decided to type the command name.

Simple enough: press ALPHA twice, type CAT, and press ENTER. Right? Wrong. That simply puts 'CAT' in Level 1 of the stack. Ah... perhaps I need to press EVAL. Nope, wrong again. Luckily, at this point I spotted the CAT key.

Manuals

My frustration grew nonetheless. In the past, when something like that happened with older HP calculators, I turned to the manuals. You could rely on them to have the information you needed. If memory serves, the HP-67 had a rather thick manual chock full of examples and printed in full color.

Heck, even my lowly 32S's manual is better than the thing I got with the 49G. For the so-called HP-49G User's Guide isn't such. It's ludicrously brief, describing only a rather small fraction of the features of the machine, with hardly any examples at all.

In my opinion, HP has done a disservice by not providing better printed manuals. Even the HP-49G Advanced User's Guide isn't any good at all. A long time ago I heard the bromide, "Manuals are written with the assumption that you already know the subject matter." I laughed back then, but now it doesn't seem so funny.

Take programming as an example. The 49G is advertised as being programmable in RPL or in BASIC. The User's Guide doesn't even mention BASIC -- in fact, there's nothing about it in the index, or in the skimpy chapter about programming.

But this chapter contains the sentence, “See the Advanced User’s Guide for information on creating and using programs in RPN mode.” Like a fool, I downloaded the Advanced manual from the Web, only to find absolutely nothing about how to write programs -- period. Not in RPN, not in BASIC -- zilch.

I never found any documentation about the BASIC programming language available on the 49G, and precious little about RPN either.

The Keyboard

It’s no news that the keyboard is plain awful. The keys are extremely hard to press, making the operation a very slow process indeed. Moreover, the complexity of the machine (see above) means a cluttered keyboard, in which it’s hard to find what you’re looking for.

Things aren’t always grouped by similarity. For example, although the bottom row contains the constants infinity and pi, the constant “i” is on the second row, almost at the other end of the keyboard.

I am aware that a machine with such a rich command set must leave many things out of the keyboard, and I’ll applaud HP’s willingness to provide menus and choose boxes. But some things need improvement -- badly. For instance, why aren’t the trigonometric functions anywhere in the MTH menu? Yes, they have keys of their own, but they ought to be in the MTH menu as well for consistency’s sake. The lack of consistency is one of the things that makes the 49G so hard to understand and use.

Still looking for the trig functions, I selected option 5 (“Real...”) from the MTH menu, which brought me to the REAL menu. This delightful little menu, with 19 options, not only doesn’t contain the trig functions, but what functions it contains are in no apparent order. Would it have been too much to ask, to put them in alphabetical sequence?

Command Names

The problem of the 49G’s complexity is exacerbated by an apparent lack of a naming convention for commands. For instance, CON and CONST both have something to do with constants. I think CON creates a matrix of constants, while CONST displays a constant -- or is it the other way around? I executed the latter with an empty stack just to see if I could figure things out, but the error message I got (“Too few arguments”) didn’t make me any wiser.

HP should take a good look at IBM’s AS/400 for a lesson in how to name commands sensibly. On the AS/400, command names always consist of two parts: a verb and its object. Both parts use a standardized set of 1-, 2-, or 3-letter abbreviations; once you learn CRT (“create”), DLT (“delete”), PGM (“program”) and LIB (“library”), you already know 4 commands: CRTPGM (“create program”), CRTLIB (“create library”), DLTPGM (“delete program”) and DLTLIB (“delete library”). Those 4 are actual AS/400 commands.

What’s important is the clarity of such commands; their names clearly suggest their function. It is of no importance that some are as long as 10 characters.

Throw-Away Programs

There are programs and programs. Some you write for use only once and then throw away. Others you save for future use. Most 49G programs seem to be of the first variety, since maintaining them is such a difficult proposition -- particularly if written in RPN mode.

I know that I’m talking about a religious issue here; after all, how are we going to put distance between us and the rest of the world, except by insisting on RPN? But I sincerely believe that HP has gone too far with RPN ever since the HP-28C came out.

Take conditional tests as an example. “If a=0 and (b=1 or c=2)” becomes the following sequence:

IF a 0 == b 1 == c 2 == OR AND THEN ...

Excuse me, but there's absolutely nothing intuitive about this. You may learn to write a program like that, and (stranger things have happened) you may even learn to `_read_` the program, but it's always at an effort. The net result is that, by insisting on using RPN everywhere, HP has made every program a throw-away program. It's easier (in most cases) to erase the program and start anew, than to try to figure out how to fix a program, once written.

Writing the program in BASIC might have solved things, but... see my entry above about the manuals.

The problem is compounded when you write a complicated logical expression as part of a loop construct such as WHILE. One way to remove the apparent complexity is to indent the program, but all indentation is gone forever as soon as you press ENTER. I know because I tried that.

Take a quick quiz, if you need proof. Look at a program you yourself wrote a year ago, and see if you understand it right away. Chances are you won't.

RPN is a wonderful logic system, but its usefulness is pretty much restricted to manual operations. Once you begin writing programs (which means whole expressions, either mathematical or logical), you're better off writing them in a more expressive logic system. The right tool for the job at hand! Writing programs in RPN makes as much sense as changing a flat tire with a screwdriver.

Of All Calculators...

I've had quite a few HP calculators (and several TI ones too), so I consider myself competent when comparing those I know. I started my relationship with HP with an HP-25 in 1976, quickly upgrading to an HP-67, then to a 41C, 41CX, 28C, 48GX, and 49G. I've had a few others (sometimes concurrently) such as a 16C and a 32S. On the TI side, I've had a SR-51A and a TI-89, as well as a few odd ones of various brands, including Casio and Sharp. I've also used other calculators, such as HP-65, 45, 35, 80, 21, 29C, 41CV, 9830A, SR-50A, SR-52, and TI-59.

Of all those calculators, I like the HP-41 best.

I think the HP-41 had the right balance between complexity and usefulness. It wasn't complex enough to make things difficult or intimidating, yet it was complex enough to be useful.

The stack had a fixed size, which means that you can be aware of its contents at all times; it's easier to remember what's in the stack when it has 4 registers than when it has 23. And you don't need a large set of commands to control it either, so stack manipulations remain simple.

The keys had a superior feel, the keyboard was uncluttered, the command set was much smaller. I could write a program in my HP-41 in a fraction of the time it would take me on the 49G. The more I use the 49G, the more convinced I am that it takes a rocket scientist to figure it out.

I miss the HP-41 sorely. I wish I could turn the clock back. It's a good thing I still own a 32S, or I would never get anything done.

Afterword: A while back I placed an ad in the Classifieds, asking to trade my 49G with a 41CX. Tom, thanks for trading with me.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-41 Fonts

Posted by [Ernie Malaga](#) on 27 Feb 2001, 6:09 p.m.

I have created two fonts that Museum visitors may find useful (or at least interesting).

The first font is "HP41 Display." It shows characters in the same "spidery" shapes that the HP-41C/CV/CX used to use. The second font is "HP Calc Keyboard," which lets the user create documentation, providing all the special characters commonly found in HP calculator keyboards.

Both fonts are for Macintosh, but they can be easily converted to Windows using a conversion utility downloadable from hardware/freeware Web sites.

I am enclosing both fonts in two formats: (a) A StuffIt archive for Macintosh users, and (b) A Zip archive for Windows users. Each archive file contains the fonts in Mac form, a jpeg showing off some features of the fonts and a brief ReadMe text file.

[hp_fonts.sit](#)

[hp_fonts.zip](#)

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP46 Printer Repair Tip

Posted by [Alex Knight](#) on 26 Feb 2001, 3:43 p.m.

I recently debugged a problem with a HP46 desktop calculator printing incorrectly and thought I would share this information. The symptom was that while the calculator would occasionally print a number correctly, more often than not the machine would print the wrong digits, or double-strike, or skip digits - it was not at all predictable as to what the machine would print. Attempting to do a LIST command to dump all the registers resulted in mostly garbage being printed.

I isolated the problem to the +15 Volt power supply. There is a 3000 microfarad, 30 Volt DC rated filter capacitor in the calculator's +15 Volt power supply that appears to provide the printer with power. In my unit, this capacitor had failed, causing a large amount of ripple on the +15V supply, which was confusing the printer and causing it to print incorrectly. Replacing this capacitor completely fixed the problem. Unfortunately, checking for this problem requires disassembly of the calculator. If you decide to attempt to repair the machine, do so at your own risk, I specifically disclaim any responsibility (and so does the MoHPC, I'm sure - sorry but my lawyer always makes me say this!). Also, please take precautions that you don't damage the machine with static electricity since you will have the covers off.

In order to get to the test point for the +15 Volt supply, you have to remove five recessed phillips-head screws on the bottom of the calculator, and then ease the top cover off of the machine. It is also necessary to remove the bottom metal cover, there are a bunch of phillips-head screws holding it on. Next, there is a brown plastic rectangular "cup" to the right of the printer mechanism (I think this is a place that a service card, or maybe even a quick reference card, could be stowed). You need to remove this cup to access the test point. There are two small phillips-head screws on the bottom of the machine that hold it in place (which is really the only reason you had to remove the bottom cover to start with).

The only way to test the power supply is to now power up the machine with all these covers removed, so be very, very careful as there is 115 Volts AC running around inside, as well as other DC voltages. It is best, I think, to have the machine sitting on a non-metal bench to do this part of the test (just make sure you don't zap the machine with static electricity). Power up the machine, and look for the test point (a small gold plated post) marked +15V in the area from which you removed the rectangular cup. You'll need to have a digital volt meter set to the appropriate scale to measure 15 Volts. If possible, use test clips instead of probes to free your hands to operate the machine. Clip the positive lead of the voltmeter to the +15V test point. Clip the negative lead of the voltmeter to a ground - the easiest place I found was on the opposite side of the small silver cylindrical tantalum capacitor whose positive side is connected to a PC track immediately to the right of the +15V test point.

With the digital volt meter hooked up, you should read something close to +15 Volts when the printer is running but not printing. Press the LIST button on the machine, and the voltage should not change much if at all. If, however, you read something a volt or so below +15 Volts when the printer is not printing, and the voltage sags while the machine is printing, then the filter capacitor may need to be replaced. On my machine, the voltmeter was reading about +13.5 Volts before printing, and sagging to as little as +11.5 Volts while printing. I then hooked up an oscilloscope and noticed that the filter capacitor was hardly providing any filtering. The 3000 microfarad, 30 Volt capacitor is the tall, silver-colored one that is immediately in front of the big power transformer in the right rear corner. If you need to replace the capacitor, try and find one that is closely matched. Don't go any lower than a 30 Volt rated capacitor, or

lower than 3000 microfarads. It isn't good practice to use a capacitor with a voltage rating that is a lot higher than necessary. I used a 3300 microfarad, 35 Volt capacitor in mine, but your Mileage may Vary.

I suspect that the capacitor went bad because this particular machine was government surplus and may have been sitting in a warehouse unused for many years. Electrolytic capacitors can go bad from nonuse, and I believe this to be the case here. Once I replaced this capacitor, the +15 Volt supply was very stable even with the printer dumping all of the registers of the machine.

I currently have a photo on my website (look in the HP section) of a HP46 with the top cover removed, you can see the transformer and the filter capacitor location in that photo. I currently don't have a view that shows the printed circuit board with the printer "cup" removed, but hope to have that available in the near future.

Good luck!

Alex Knight

The Calculator Museum Web Page

<http://www.calcmuseum.com>

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

Opening the HP48: Addendum

Posted by [David Fenyes](#) on 20 Feb 2001, 8:31 p.m.

I left out a couple of points:

- 1) (Thanks to Nick Zacharopoulos) To make the reassembled calculator super-sturdy, completely remove the two plastic rivet-posts from above the display, and glue little 2-56 plastic hex nuts at their base. Drill out the portion of the posts attached to the calculator back with a small bit, large enough to pass plastic 2-56 bolts. Cut two 2-56 plastic screws just long enough to lock into the hex nuts with the case closed. Once the glue dries, close the case, and tighten down the screws. The calculator will be very solid.
- 2) If you plan to make your own serial cable, you may want to consider clipping off the serial connector, and soldering+epoxying a 4-pin 0.100" molex connector instead (it will fit). These are a little sturdier, and it's easy to find the female mates. I've done this and it works well. Note that unlike the other mods, this will be very evident on inspection.

David.

Edited: 20 July 2003, 5:32 p.m.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

Go back to the main exhibit hall

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP48 Disassembly (from the back)

Posted by [David Fenyes](#) on 18 Feb 2001, 3:00 p.m.

I noticed the comment in the HP Museum disassembly notes, that you can't disassemble the 48's without serious damage. I posted this to comp.sys.hp48 a while back (1995 according to Adrian Drury), and thought I'd repost it, with a couple of extra observations. Adrian Drury took pictures as he disassembled his '48 based on these instructions, and has made an excellent web page, which I recommend you visit:

<http://www.contrib.andrew.cmu.edu/~drury/newhp/contents.html>

This article does contain some useful points not mentioned in the web page, most importantly how to secure the top of the calculator with small plastic bolts (Thanks to Nick Zacharopoulos).

David.

Hello fellow HP48 owners :-)

If you have a 48S/G, you may have thought about expanding your memory. If you refrained because you didn't want to uglify your calculator, now you have no excuse. I bought two 48S's from EduCalc, and disassembled the first one according to Paul Smith's article (thanks for your bravery, Paul:-) I managed to get it back together, using epoxy to secure the faceplate down with a minimum of wrinkling. My first new 288K calc is quite usable, but looks like an old battle implement (at least a little beat up :-). It's safe to say that a calculator disassembled this way will never be the same again. I wanted calc#2 to be prettier. It turns out that it's relatively easy to make cuts in places not normally visible (behind the red IR port lens & in the battery compartment) to render the front and back halves separable. The benefits of this approach are that your 48 still looks like new, and the structural columns can be salvaged to retain the solid "HP" feel of the new calc. I've written up the directions below.

Disclaimer: The directions are based on MY 2 HP48S's, and may not apply to any others in the world. I'm providing this info so that someone (maybe you) may benefit from it. I can't be responsible for any damage you inflict on anything (including yourself) by following these directions. I also can't guarantee the accuracy of my descriptions. Please THINK about what you're doing to make sure it's not obviously insane (loosening the LCD with a hammer, etc.) You will void your warranty if you take apart the calculator.

Disclaimer softener: This worked for me :-) If you follow directions, you and your 48 should be OK. Also, this is a very long post describing a very simple procedure. Don't be discouraged by the length. The detail is there to help prevent you from goofing :-)

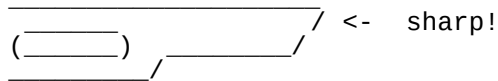
Good luck, and enjoy!

David. -- David Fenyes dave@fenyes.net

Tools & Materials

Feel free to improvise. I used:

- 1) an X-acto knife, preferably with a #6 blade (thin, narrow trapezoidal blade with a shallow angle -->



The blade should fit nicely in those keyboard template slots at the ON, ENTER, MTH, NXT, <-, and + keys.

- 2) A broad, blunt, thin blade, such as the screwdriver on a Victorinox-brand swiss army knife. A butterknife may work. Using a narrow, thick screwdriver will leave ugly marks in the soft plastic.

- 3) A drill with a very small bit, or a 1/4" bit.

- 4) The slick backing paper from a sticker

- 5) a couple of butter knives, Q-tip stems, or wooden matchsticks.

- 6) Some 14 Ga solid copper wire (I used laquered magnet wire).

Directions:

Top - toward the IR lens/serial port connector. Bottom - toward the "ON 0 . SPC +" row of keys Back - toward the battery-compartment side of the calc. Front - the faceplate side of the calc.

Overview:

=====

Instead of ripping off the tin faceplate and destroying the rivet heads, you will make well placed incisions behind the IR lens and in the battery compartment to defeat the 10 plastic rivets. The abbreviated plan is:

- 1) empty battery compartment and remove IR lens.
- 2) make 2 holes in the recess behind the IR lens in order to gain access to the 4 rivets above the display, and slice through them.
- 3) Enlarge the hole behind the (+) battery terminal, and slice the exposed plastic rivet. Make a corresponding hole and slice the corresponding rivet on the right side of the compartment.
- 4) Dig out the bases of the remaining four rivets beneath the bottom battery-separator ridge in the compartment.
- 5) Separate the two halves of the calculator.
- 6) "repair" the plastic rivets to restore structural integrity.

Detailed Instructions:

=====

0) back up your memory and turn the calculator off.

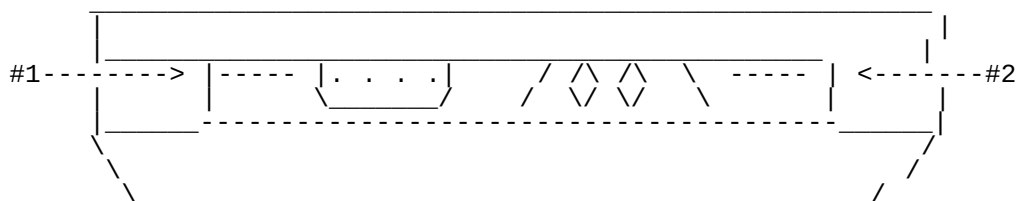
1) Remove the battery case cover and batteries. Carefully peel off the foam pads/battery connectors from the battery compartment and stick them on the piece of sticker backing.

2) Remove the dark red plastic IR port cover:

- Turn the calc back facing you, wedge the dull blade between the IR lens and the calc body about 1cm from the left edge and pry the lens off. Repeat for the right side. The lens will pop out unharmed. (Or you can use your fingernails to pull the lens off for minimal risk of blemishes)

3) Begin the "plastic surgery":

- Look into the recess behind the IR lens:



I have drawn the IR and serial ports, and the two slots you will need to make. Look into the serial connector and note the level of the PCB. Using the X-acto knife, make a 1cm horizontal score (#1) about halfway between the level of the serial pins and the PCB.

Use a drill with a very small bit (~1mm) to make a row of holes along the scored line. *REMEMBER* where the PCB is, and be sure to angle the drill very slightly downward. If you do this, you will be drilling into air, and will not harm anything of consequence inside the case. Using the X-acto blade, connect the holes to make a slot. If you can't see very well inside the slot, have faith. Stick the X-acto blade into the slot, angled slightly down (away from the PCB), and slice anything you encounter. You should be able to move the blade freely in the slot. To a casual observer, you will look as if you are lobotomizing your calculator. In fact your blade will only hit the 2 plastic rivet bodies behind the slot (and air).

Repeat the procedure to make a symmetrical slot on the other side (position #2) next to the IR port. make it the same width and level as the first slot. Again, you will be drilling and slicing into air if you keep the angle slightly down.

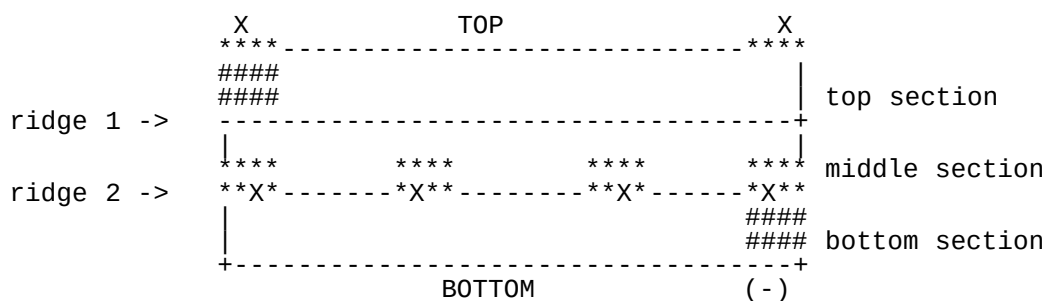
** Alternative: Your task may be considerably simpler if instead of making slots, you use a 1/4" bit to drill holes slightly below the slots indicated above (angled downward, of course,) to give yourself a good view of the rivets before you cut them.

Bravo :-) You have accomplished the scariest part. You should now be able to pry the top of the calculator apart very slightly.

4) Slice the two rivets at the top edge of the battery compartment:

Use the following schematic for the next two sections. Remember, it's ASCII art, so don't measure it with a ruler :-). Read the text for specifics:

Battery Compartment



X = plastic rivet # = the hole is already there. * = you make the hole.

Turn the calculator so its back is facing you. Bend the (+) battery terminal (the one in the top left corner of the compartment) so that it sticks straight up and is flat against the left side of the compartment. This will expose a rectangular hole in the top side of the compartment. Using the blade, slice upward along the edges of the hole and across to extend the hole about 3-4 mm toward the rear of the calculator.

The hole is now a window into the calc innards. Look into the window at a shallow angle. You will see a plastic rivet. This is the rivet amidst the "[6] [X] [9] [/]" keys. You can slice this easily with the blade.

Now you need to make an identical window in the top right corner of the case and slice through the second rivet (the "[Lshift] [Alpha] [4] [7]" rivet.)

5) Free the remaining four rivets from their bases:

The battery compartment is divided into three sections by two ridges. The four remaining rivets are behind the bottom ridge, very slightly toward the middle section. They lie amidst the following keys:

```

"{1} {ON} {0} {Rshift}"
"{1} {2} {0} {."}"
"{2} {3} {.} {SPC}"
"{3} {-} {SPC} {+}"
  
```

You will harm nothing by digging around, but you want to damage as little of the compartment as possible. Start slicing out sections of the bottom ridge using the keys as a guide for the horizontal dig sites along the ridge. Actually, the "+" signs on the bottom and middle battery icons accurately indicate the positions of the middle two rivets. When you encounter a small (~14Ga) circular hole, you have identified a rivet. You need to free each rivet from the surrounding plastic so that it wobbles freely.

HINT: Since you'll be doing lots of digging, you may try to use the edges of the battery case for leverage. If you line the externally visible outline of the compartment with masking tape, you'll prevent cosmetic damage.

6) Separate the two halves of the case:

Now the calculator's keyboard should be facing you. Insert the blade in the keyboard-template slot near the NXT key. The blade should go straight back to the back half of the calc. Apply pressure on the handle toward the midline of the

calc. This will disengage the front half from the back half. With the screwdriver blade, gently pry the halves apart at the [F] key. Place something (like a butterknife) between the halves to keep them apart. Repeat the procedure at the [MTH] key (pry near [A]), the [ENTER] key (pry near [SIN]), and the [backspace] key (pry near [1/x]). The whole top part of the calculator should be nicely separated. The bottom 3 hooks are somewhat more resistant to separation, probably because of their close proximity to one another. The best thing to do here is insert the blunt blade (victorinox screwdriver or butter knife) between the two halves near the [-] key. Pull the handle up toward yourself. The bottom half should disengage from the top half. Stick something (like the butterknife or wooden matchstick) between the halves to keep them apart. Repeat this now at the bottom of the calc (the "." key) The calculator should completely come apart now.

7) Putting it back together.

You could probably just snap the two halves together. However, if you want a little more stability, it is possible to restore the structural function of the rivets.

The rivets are hollow. Using the X-acto blade, clean up the ends of the rivets on both halves of the calculator. The hollow center of each rivet should be round and free of plastic debris. You may also wish to clean up your hacking near the IR port and in the battery compartment.

14 Ga laquered solid wire fits snugly in the hollow rivet bodies. Cut six straight pieces of the solid wire, exactly 1cm each. Make sure that the ends of each piece fit neatly into both halves of the top six plastic rivets (you won't need to restore the four bottom rivets that you freed from their bases.) Push the wire pieces into the top six rivet bodies in the back half of the calculator.

Bend the (+) battery terminal back into shape. Align the front half of the calculator carefully with the back half. The rivet remnants on the front half should align with the wire pieces sticking up from the bottom half. The (+) and (-) battery terminals should fit through their respective slots, and the top half should rest flat on the bottom half. Press the calculator together starting at the bottom. The pieces should snap together with no difficulty. If the rivets and wire are misaligned, you may need to push the wire into alignment with a needle. Remove the two foam-backed battery terminals from the sticker-backing and replace them in the compartment. Replace the batteries and battery compartment cover. Snap the IR lens back into place.

Congratulations! You have disassembled (and hopefully upgraded :-) your calculator without ruining its appearance :-) If you were careful, it should look brand-new!

Edited: 20 July 2003, 5:32 p.m.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

Serial port downloads from (Linux) PC to HP-41

Posted by [Tony Duell](#) on 10 Feb 2001, 2:59 p.m.

Here's my file on downloading from a (linux) PC to an HP41 using the serial interface and extended I/O module. You'll have to modify the PC configuration instructions for your OS, but I can't tell you how to do it...

Transferring progrmas between an HP41 and a PC using a serial interface

This document describes how to use the HP82164 HPIL-RS232 interface to connect an HP41 calculator (fitted with the HP82160 HPIL interface and the Extended I/O module) to a Linux PC, and to transfer programs in both directions between them.

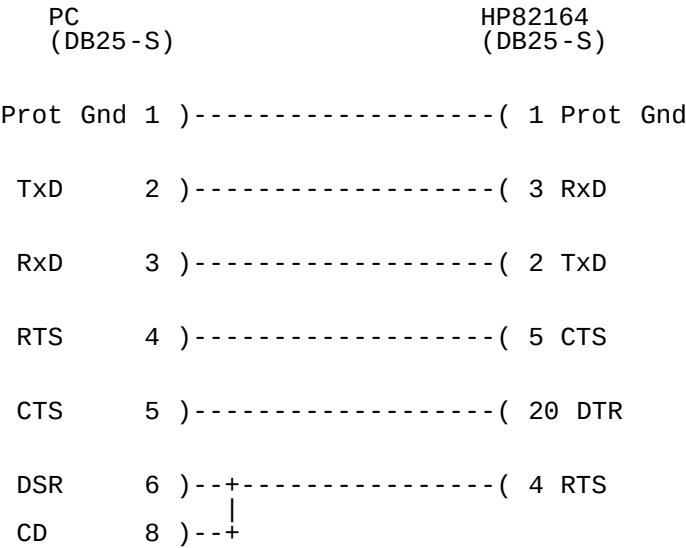
Throughout this document, the PC's serial port will be named /dev/cua0 . Change this as appropriate, of course.

Hardware connections

Connect the HPIL side of the HP82164 to the HP41's IL module in the usual way.

The jumper plug inside the HP82164 interface should be set to the DTE position. See the interface's user manual for more details of this.

Connect the HP82164's RS232 port to the PC using a standard PC modem cable and a special adapter. The latter is needed owing to the rather strange hardware handshaking of the HP82164, and the following diagram gives the necessary wiring :



```

Sig Gnd 7 )------( 7 Sig Gnd

DTR      20 )-----+---( 6 DSR
                  |
                  +---( 8 CD

```

Configuring the interfaces

On the PC, turn off buffering and enable hardware handshaking by typing :

```
stty raw crtscts </dev/cua0
```

On the HP41, load the SETRS program and run it to set the HP82164 interface to hardware handshake mode and to make it the primary device. A binary file of this program is given in hardware/setrs.p41 in the lif_utils distribution, and is given as a listing below :

```

0001  LBL "SETRS"
0002  66
0003  FINDAID
0004  SELECT
0005  CF 17
0006  REMOTE
0007  "SL2"
0008  OUTA
0009  "LI5"
0010  OUTA
0011  "C0"
0012  OUTA
0013  NOTREM
0014  END

```

Transferring a program from the HP41 to the PC

On the PC type :

```
inp41 < /dev/cua0 >program_file
```

where program_file is the filename that you want the binary program stored in. Alternatively you can pipe the output of inp41 to prog41 to get a text listing by typing :

```
inp41 < /dev/cua0 | prog41
```

On the HP41:

Put the name of the program to send (or more exactly one of the global labels in that program) in the alpha register.

Then execute : OUTP

When the program has transfered, the PC should display

checksum good

and the program will be stored in the specified file

Transferring a program from the PC to the HP41

On the PC type :

```
outp41 < program_file >/dev/cua0
```

where program_file is the name of a file containing an HP41 program.

The input to outp41 can be piped from lifget (for example) in the standard way to transfer a file from a lif disk image. For example :

```
lifget -f disk1 PROG | outp41 > /dev/cua0
```

will transfer the file PROG from the lif disk image disk1.

On the HP41:

GTO .. to pack program memory and create a dummy program at the END.

then execute : INP

When the program has transfered, the X-register display will return. CAT 1 to check that the program has loaded.

A. R. Duell, 2-Oct-2000.

This file is part of the LIF Utilities distribution and has been placed under the GNU Public License version 2.0

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP 65 card reader tips: reinforcing the worm gear clutch

Posted by [Rafa, Spain](#) on 28 Jan 2001, 3:55 p.m.

The following tips were very useful when repairing my HP-65 card reader. Some I have discover, and some I've learned from others. I've posted them here for anyone that could be interested. I think they will serve for other HP models too.

- Reinforcing the worm gear clutch: To do this I needed a very sharpened scissors and a piece of plastic film. (I don't know the proper name in English, but it is the one that is transparent and is used in the kitchen to wrap aliments, cover bowls with food, etc. –the one that sticks everywhere due to static electricity). First separate the worm gear form the motor and take a look at the little shaft that comes out of the motor. The idea is to cover this shaft with the plastic so that when the worm gear is attached again there will be more pressure and the clutch will be reinforced. Cut the plastic to make a very little rectangular piece of about 14x7 mm (patience is needed!). Fold it from the middle to make a 7x7 mm square and then cut the four corners to give it a rounded shape. Now you can see if it is the proper size by covering the motor shaft with it. If it is too large you may make it shorter, and if it is too short that covers only a part of the shaft, try to do a larger one. When you get the right one just cover again the shaft and while holding it to keep it centred reattach the worm gear to the bottom. You will need to make pressure, but be careful not to twist the shaft!. If you did it the right size no traces of plastic should be seen. (note that if the clutch is still worn you can fold the plastic two times instead of one (or zero)). I suppose this procedure is better than gluing the worm gear because the motor is able to slip in case the card reader gets stuck. And of course you can always reverse the fix.

- Putting oil: I have improved the card reader functioning by putting a little amount of oil in some strategic places. I have eliminated bad noises too. The worm gear may actually get stuck if there is some dirty in the plastic hole where it is attached (the opposite end to the motor). I've get good results putting oil there (just a little amount). Another place is inside the rubber drive wheel shaft –it must be a very little quantity or it could pass to the rubber causing the cards to slip. Finally I greatly reduced the noise of the motor by putting just a single drop of oil (caught in the head of a needle) where the motor shaft comes out (red plastic piece there). I don't know if it was a tremendous error but well, the motor rolls now very smooth and silent!.

The motor of course vibrates when functioning. If it is not well attached to the plastic structure the vibration noises are very awful. I've make this noise 'cleaner' by tightening a bit more the two screws that hold the motor to the plastic structure so that motor and plastic vibrate as a single piece.

All this tips have prove successful for me, I hope they will be the same for you!

Thanks for reading

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[Return to the Index]

[Previous | Next]

The (not so) Big List of HP Part Numbers

Posted by [John Kono](#) on 5 Jan 2001, 4:27 a.m.

[It's late and I'm bored, so here is the first installment of what I hope will be a (really) Big List of part numbers for old HP books and accessories.]

Voyager Series

	Part Number	
	Retail	"5 by 5"*
Accessories	-----	-----
	82174A	82174-60001 Vinyl Case
	92169A	Leather Case
	92169B	- Black
	92169C	- Brown
	92177S	- Bugundy
		Acrylic Desktop Stand
		HP-11C Owner's Handbook
	00011-90001	- English
	00011-90002	- German
	00011-90003	- French
	00011-90004	- Italian
	00011-90005	- Spanish
	00011-90006	- Swedish
	00011-90008	- Japanese
	00011-90017	- Dutch
	00011-90018	- Danish
	92233F	HP-12C Quick Reference Book - English
	00012-90022	HP-12C Training Guide - English
		HP-12C Owner's Handbook
	00012-90001	- English
	00012-90002	- German
	00012-90003	- French
	00012-90004	- Italian
	00012-90005	- Spanish
	00012-90006	- Swedish
	00012-90008	- Japanese
	00012-90013	- Dutch
	00012-90014	- Danish
		HP-15C Owner's Handbook
	00015-90001	- English
	00015-90002	- German
	00015-90003	- French
	00015-90004	- Italian
	00015-90005	- Spanish
	00015-90006	- Swedish
	00015-90008	- Japanese
	00015-90009	- Dutch
	00015-90010	- Danish
		HP-16C Owner's Handbook
	00016-90001	- English
	00016-90002	- German
	00016-90003	- French
	00016-90005	- Japanese

Application Books		00011-90009	HP-11C Solutions Handbook - English
		00012-90009	HP-12C Solutions Handbook - English
		00012-90015	HP-12C Real Estate Applications - English
		00012-90021	HP-12C Leasing Applications - English
			HP-15C Advanced Functions Handbook
		00015-90011	- English
		00015-90013	- German
		00015-90014	- French
	92234S		An Easy Course in Using the HP-12C - English
	92234W		An Easy Course in Programming the HP-11C and HP-
15C - English			

HP-41C/CV/CX

[I have chosen to start this section with the Solutions Books simply because I noticed the the HP-41 page on this site doesn't list part numbers for these. I will complete the HP-41 Accessories at a later date. -jk]

Solutions Books	Part Number		"5 by 5"*
	Retail		
	-----		-----
	92234T		An Easy Course in Programming the HP-41
	92235S		Electrical Circuits for Students
	92235R		Statistics for Students
		00041-90093	Antennas
		00041-90094	Business Statistics/Marketing/Sales
		00041-90145	Calendars
		00041-90100	Chemical Engineering
		00041-90102	Chemistry
		00041-90089	Civil Engineering
		00041-90092	Control Systems
		00041-90088	Electrical Engineering
		00041-90139	Fluid Dynamics & Hydraulics
		00041-90099	Games
		00041-90443	Games II
		00041-90084	Geometry
		00041-90140	Heating, Ventilating & Air Conditioning
		00041-90083	High Level Math
		00041-90096	Home Construction Estimating
		00041-90086	Lending, Savings & Leasing
		00041-90090	Mechanical Engineering
		00041-90143	Optometry I (General)
		00041-90144	Optometry II (Contact Lens)
		00041-90142	Physics
		00041-90136	Real Estate
		00041-90137	Small Business
		00041-90138	Solar Engineering
		00041-90441	Structural Design
		00041-90141	Surveying
		00041-90082	Test Statistics
		00041-90395	Time Module Solutions I

* - The "5 by 5" numbers are currently used by HP's Parts Direct service for ordering replacement parts, though there is no guarantee that the part you want is still available. [Does anyone know if these are used only for replacement designations? -jk]

If you notice any mistakes or omissions, please send the corrections to me directly. I will include them in the next updated list. Thanks! [Note: I still have a number of sections to add before this list is complete (including HP-71, Champion, and *lots* of HP-41 items).]

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)
[\[Previous | Next \]](#)

HP Calculator Opcode Map

Posted by [George Weigt](#) on 14 Dec 2000, 5:48 p.m.

HP Calculator Opcode Map

To find the opcode for a particular instruction, or-together the coordinate bit strings, treating xxxx as 0000. This map was created with the help of [Eric Smith's](#) assembler and may be used to help understand the [HP-45 simulator](#).

	xxxx000000	xxxx000100	xxxx001000	xxxx001100	xxxx010000	xxxx010100	xxxx011000	xxxx011100
0000xxxx00	no operation	1 -> s0	.	0 -> p	select rom 0	if s0 = 0	load constant 0	p - 1 -> p
0001xxxx00	buffer -> rom address	1 -> s1	.	1 -> p	.	if s1 = 0	load constant 1	.
0010xxxx00	memory insert	1 -> s2	.	2 -> p	select rom 1	if s2 = 0	load constant 2	.
0011xxxx00	.	1 -> s3	.	3 -> p	keys -> rom address	if s3 = 0	load constant 3	.
0100xxxx00	mark and search	1 -> s4	.	4 -> p	select rom 2	if s4 = 0	load constant 4	.
0101xxxx00	.	1 -> s5	.	5 -> p	.	if s5 = 0	load constant 5	.
0110xxxx00	memory delete	1 -> s6	.	6 -> p	select rom 3	if s6 = 0	load constant 6	.
0111xxxx00	.	1 -> s7	.	7 -> p	.	if s7 = 0	load constant 7	.
1000xxxx00	rom address -> buffer	1 -> s8	.	8 -> p	select rom 4	if s8 = 0	load constant 8	.
1001xxxx00	.	1 -> s9	.	9 -> p	.	if s9 = 0	load constant 9	.
1010xxxx00	search for label	1 -> s10	.	10 -> p	select rom 5	if s10 = 0	.	.
1011xxxx00	.	1 -> s11	.	11 -> p	.	if s11 = 0	.	.
1100xxxx00	pointer advance	.	.	12 -> p	select rom 6	.	.	.
1101xxxx00	.	.	.	13 -> p	.	.	.	.

1110xxxx00	.	.	.	14 -> p	select rom 7	.	.	.
1111xxxx00	.	.	.	15 -> p	.	.	.	.

	xxxx100000	xxxx100100	xxxx101000	xxxx101100	xxxx110000	xxxx110100	xxxx111000	xxxx111100
0000xxxx00	.	0 -> s0	display toggle	if p # 0	return	clear status	.	p + 1 -> p
0001xxxx00	.	0 -> s1	.	if p # 1	.	delayed select rom 0	.	.
0010xxxx00	.	0 -> s2	c exchange m	if p # 2	.	.	.	.
0011xxxx00	.	0 -> s3	.	if p # 3	.	delayed select rom 1	.	.
0100xxxx00	.	0 -> s4	c -> stack	if p # 4	.	.	.	.
0101xxxx00	.	0 -> s5	.	if p # 5	.	delayed select rom 2	.	.
0110xxxx00	.	0 -> s6	stack -> a	if p # 6	.	.	.	.
0111xxxx00	.	0 -> s7	.	if p # 7	.	delayed select rom 3	.	.
1000xxxx00	.	0 -> s8	display off	if p # 8	.	delayed select group 0	.	.
1001xxxx00	.	0 -> s9	.	if p # 9	c -> data address	delayed select rom 4	.	.
1010xxxx00	.	0 -> s10	m -> c	if p # 10	.	delayed select group 1	.	.
1011xxxx00	.	0 -> s11	.	if p # 11	c -> data	delayed select rom 5	data -> c	.
1100xxxx00	.	.	down rotate	if p # 12	.	.	.	.
1101xxxx00	.	.	.	if p # 13	.	delayed select rom 6	.	.
1110xxxx00	.	.	clear registers	if p # 14	.	.	.	.
1111xxxx00	.	.	.	if p # 15	.	delayed select rom 7	.	.

	xxxx000010	xxxx000110	xxxx001010	xxxx001110	xxxx010010	xxxx010110	xxxx011010	xxxx011110
0000xxxx10	if b[p] = 0	if b[m] = 0	if b[x] = 0	if b[w] = 0	if b[wp] = 0	if b[ms] = 0	if b[xs] = 0	if b[s] = 0
0001xxxx10	if a >= c[p]	if a >= c[m]	if a >= c[x]	if a >= c[w]	if a >= c[wp]	if a >= c[ms]	if a >= c[xs]	if a >= c[s]
0010xxxx10	b -> c[p]	b -> c[m]	b -> c[x]	b -> c[w]	b -> c[wp]	b -> c[ms]	b -> c[xs]	b -> c[s]
0011xxxx10	0 -> c[p]	0 -> c[m]	0 -> c[x]	0 -> c[w]	0 -> c[wp]	0 -> c[ms]	0 -> c[xs]	0 -> c[s]
0100xxxx10	shift left a[p]	shift left a[m]	shift left a[x]	shift left a[w]	shift left a[wp]	shift left a[ms]	shift left a[xs]	shift left a[s]
0101xxxx10	a - c -> c[p]	a - c ->	a - c -> c[x]	a - c ->	a - c ->	a - c ->	a - c ->	a - c -> c[s]

		c[m]		c[w]	c[wp]	c[ms]	c[xs]	
0110xxxx10	c -> a[p]	c -> a[m]	c -> a[x]	c -> a[w]	c -> a[wp]	c -> a[ms]	c -> a[xs]	c -> a[s]
0111xxxx10	a + c -> c[p]	a + c -> c[m]	a + c -> c[x]	a + c -> c[w]	a + c -> c[wp]	a + c -> c[ms]	a + c -> c[xs]	a + c -> c[s]
1000xxxx10	if a >= b[p]	if a >= b[m]	if a >= b[x]	if a >= b[w]	if a >= b[wp]	if a >= b[ms]	if a >= b[xs]	if a >= b[s]
1001xxxx10	shift right c[p]	shift right c[m]	shift right c[x]	shift right c[w]	shift right c[wp]	shift right c[ms]	shift right c[xs]	shift right c[s]
1010xxxx10	shift right b[p]	shift right b[m]	shift right b[x]	shift right b[w]	shift right b[wp]	shift right b[ms]	shift right b[xs]	shift right b[s]
1011xxxx10	shift right a[p]	shift right a[m]	shift right a[x]	shift right a[w]	shift right a[wp]	shift right a[ms]	shift right a[xs]	shift right a[s]
1100xxxx10	a - b -> a[p]	a - b -> a[m]	a - b -> a[x]	a - b -> a[w]	a - b -> a[wp]	a - b -> a[ms]	a - b -> a[xs]	a - b -> a[s]
1101xxxx10	a - c -> a[p]	a - c -> a[m]	a - c -> a[x]	a - c -> a[w]	a - c -> a[wp]	a - c -> a[ms]	a - c -> a[xs]	a - c -> a[s]
1110xxxx10	a + b -> a[p]	a + b -> a[m]	a + b -> a[x]	a + b -> a[w]	a + b -> a[wp]	a + b -> a[ms]	a + b -> a[xs]	a + b -> a[s]
1111xxxx10	a + c -> a[p]	a + c -> a[m]	a + c -> a[x]	a + c -> a[w]	a + c -> a[wp]	a + c -> a[ms]	a + c -> a[xs]	a + c -> a[s]

	xxxx100010	xxxx100110	xxxx101010	xxxx101110	xxxx110010	xxxx110110	xxxx111010	xxxx111110
0000xxxx10	0 -> b[p]	0 -> b[m]	0 -> b[x]	0 -> b[w]	0 -> b[wp]	0 -> b[ms]	0 -> b[xs]	0 -> b[s]
0001xxxx10	if c[p] >= 1	if c[m] >= 1	if c[x] >= 1	if c[w] >= 1	if c[wp] >= 1	if c[ms] >= 1	if c[xs] >= 1	if c[s] >= 1
0010xxxx10	0 - c -> c[p]	0 - c -> c[m]	0 - c -> c[x]	0 - c -> c[w]	0 - c -> c[wp]	0 - c -> c[ms]	0 - c -> c[xs]	0 - c -> c[s]
0011xxxx10	0 - c - 1 -> c[p]	0 - c - 1 -> c[m]	0 - c - 1 -> c[x]	0 - c - 1 -> c[w]	0 - c - 1 -> c[wp]	0 - c - 1 -> c[ms]	0 - c - 1 -> c[xs]	0 - c - 1 -> c[s]
0100xxxx10	a -> b[p]	a -> b[m]	a -> b[x]	a -> b[w]	a -> b[wp]	a -> b[ms]	a -> b[xs]	a -> b[s]
0101xxxx10	c - 1 -> c[p]	c - 1 -> c[m]	c - 1 -> c[x]	c - 1 -> c[w]	c - 1 -> c[wp]	c - 1 -> c[ms]	c - 1 -> c[xs]	c - 1 -> c[s]
0110xxxx10	if c[p] = 0	if c[m] = 0	if c[x] = 0	if c[w] = 0	if c[wp] = 0	if c[ms] = 0	if c[xs] = 0	if c[s] = 0
0111xxxx10	c + 1 -> c[p]	c + 1 -> c[m]	c + 1 -> c[x]	c + 1 -> c[w]	c + 1 -> c[wp]	c + 1 -> c[ms]	c + 1 -> c[xs]	c + 1 -> c[s]
1000xxxx10	b exchange c[p]	b exchange c[m]	b exchange c[x]	b exchange c[w]	b exchange c[wp]	b exchange c[ms]	b exchange c[xs]	b exchange c[s]
1001xxxx10	if a[p] >= 1	if a[m] >= 1	if a[x] >= 1	if a[w] >= 1	if a[wp] >= 1	if a[ms] >= 1	if a[xs] >= 1	if a[s] >= 1
1010xxxx10	c + c -> c[p]	c + c -> c[m]	c + c -> c[x]	c + c -> c[w]	c + c -> c[wp]	c + c -> c[ms]	c + c -> c[xs]	c + c -> c[s]
1011xxxx10	0 -> a[p]	0 -> a[m]	0 -> a[x]	0 -> a[w]	0 -> a[wp]	0 -> a[ms]	0 -> a[xs]	0 -> a[s]
1100xxxx10	a exchange b[p]	a exchange b[m]	a exchange b[x]	a exchange b[w]	a exchange b[wp]	a exchange b[ms]	a exchange b[xs]	a exchange b[s]
1101xxxx10	a - 1 -> a[p]	a - 1 ->	a - 1 -> a[x]	a - 1 ->	a - 1 ->	a - 1 ->	a - 1 ->	a - 1 -> a[s]

		a[m]		a[w]	a[wp]	a[ms]	a[xs]	
1110xxxx10	a exchange c[p]	a exchange c[m]	a exchange c[x]	a exchange c[w]	a exchange c[wp]	a exchange c[ms]	a exchange c[xs]	a exchange c[s]
1111xxxx10	a + 1 -> a[p]	a + 1 -> a[m]	a + 1 -> a[x]	a + 1 -> a[w]	a + 1 -> a[wp]	a + 1 -> a[ms]	a + 1 -> a[xs]	a + 1 -> a[s]

	xxxx000001	xxxx000101	xxxx001001	xxxx001101	xxxx010001	xxxx010101	xxxx011001	xxxx011101
0000xxxx01	jsb 0	jsb 1	jsb 2	jsb 3	jsb 4	jsb 5	jsb 6	jsb 7
0001xxxx01	jsb 16	jsb 17	jsb 18	jsb 19	jsb 20	jsb 21	jsb 22	jsb 23
0010xxxx01	jsb 32	jsb 33	jsb 34	jsb 35	jsb 36	jsb 37	jsb 38	jsb 39
0011xxxx01	jsb 48	jsb 49	jsb 50	jsb 51	jsb 52	jsb 53	jsb 54	jsb 55
0100xxxx01	jsb 64	jsb 65	jsb 66	jsb 67	jsb 68	jsb 69	jsb 70	jsb 71
0101xxxx01	jsb 80	jsb 81	jsb 82	jsb 83	jsb 84	jsb 85	jsb 86	jsb 87
0110xxxx01	jsb 96	jsb 97	jsb 98	jsb 99	jsb 100	jsb 101	jsb 102	jsb 103
0111xxxx01	jsb 112	jsb 113	jsb 114	jsb 115	jsb 116	jsb 117	jsb 118	jsb 119
1000xxxx01	jsb 128	jsb 129	jsb 130	jsb 131	jsb 132	jsb 133	jsb 134	jsb 135
1001xxxx01	jsb 144	jsb 145	jsb 146	jsb 147	jsb 148	jsb 149	jsb 150	jsb 151
1010xxxx01	jsb 160	jsb 161	jsb 162	jsb 163	jsb 164	jsb 165	jsb 166	jsb 167
1011xxxx01	jsb 176	jsb 177	jsb 178	jsb 179	jsb 180	jsb 181	jsb 182	jsb 183
1100xxxx01	jsb 192	jsb 193	jsb 194	jsb 195	jsb 196	jsb 197	jsb 198	jsb 199
1101xxxx01	jsb 208	jsb 209	jsb 210	jsb 211	jsb 212	jsb 213	jsb 214	jsb 215
1110xxxx01	jsb 224	jsb 225	jsb 226	jsb 227	jsb 228	jsb 229	jsb 230	jsb 231
1111xxxx01	jsb 240	jsb 241	jsb 242	jsb 243	jsb 244	jsb 245	jsb 246	jsb 247

	xxxx100001	xxxx100101	xxxx101001	xxxx101101	xxxx110001	xxxx110101	xxxx111001	xxxx111101
0000xxxx01	jsb 8	jsb 9	jsb 10	jsb 11	jsb 12	jsb 13	jsb 14	jsb 15
0001xxxx01	jsb 24	jsb 25	jsb 26	jsb 27	jsb 28	jsb 29	jsb 30	jsb 31
0010xxxx01	jsb 40	jsb 41	jsb 42	jsb 43	jsb 44	jsb 45	jsb 46	jsb 47
0011xxxx01	jsb 56	jsb 57	jsb 58	jsb 59	jsb 60	jsb 61	jsb 62	jsb 63
0100xxxx01	jsb 72	jsb 73	jsb 74	jsb 75	jsb 76	jsb 77	jsb 78	jsb 79
0101xxxx01	jsb 88	jsb 89	jsb 90	jsb 91	jsb 92	jsb 93	jsb 94	jsb 95
0110xxxx01	jsb 104	jsb 105	jsb 106	jsb 107	jsb 108	jsb 109	jsb 110	jsb 111
0111xxxx01	jsb 120	jsb 121	jsb 122	jsb 123	jsb 124	jsb 125	jsb 126	jsb 127
1000xxxx01	jsb 136	jsb 137	jsb 138	jsb 139	jsb 140	jsb 141	jsb 142	jsb 143
1001xxxx01	jsb 152	jsb 153	jsb 154	jsb 155	jsb 156	jsb 157	jsb 158	jsb 159
1010xxxx01	jsb 168	jsb 169	jsb 170	jsb 171	jsb 172	jsb 173	jsb 174	jsb 175
1011xxxx01	jsb 184	jsb 185	jsb 186	jsb 187	jsb 188	jsb 189	jsb 190	jsb 191
1100xxxx01	jsb 200	jsb 201	jsb 202	jsb 203	jsb 204	jsb 205	jsb 206	jsb 207
1101xxxx01	jsb 216	jsb 217	jsb 218	jsb 219	jsb 220	jsb 221	jsb 222	jsb 223
1110xxxx01	jsb 232	jsb 233	jsb 234	jsb 235	jsb 236	jsb 237	jsb 238	jsb 239
1111xxxx01	jsb 248	jsb 249	jsb 250	jsb 251	jsb 252	jsb 253	jsb 254	jsb 255

	xxxx000011	xxxx000111	xxxx001011	xxxx001111	xxxx010011	xxxx010111	xxxx011011	xxxx011111
0000xxxx11	go to 0	go to 1	go to 2	go to 3	go to 4	go to 5	go to 6	go to 7
0001xxxx11	go to 16	go to 17	go to 18	go to 19	go to 20	go to 21	go to 22	go to 23
0010xxxx11	go to 32	go to 33	go to 34	go to 35	go to 36	go to 37	go to 38	go to 39
0011xxxx11	go to 48	go to 49	go to 50	go to 51	go to 52	go to 53	go to 54	go to 55
0100xxxx11	go to 64	go to 65	go to 66	go to 67	go to 68	go to 69	go to 70	go to 71
0101xxxx11	go to 80	go to 81	go to 82	go to 83	go to 84	go to 85	go to 86	go to 87
0110xxxx11	go to 96	go to 97	go to 98	go to 99	go to 100	go to 101	go to 102	go to 103
0111xxxx11	go to 112	go to 113	go to 114	go to 115	go to 116	go to 117	go to 118	go to 119
1000xxxx11	go to 128	go to 129	go to 130	go to 131	go to 132	go to 133	go to 134	go to 135
1001xxxx11	go to 144	go to 145	go to 146	go to 147	go to 148	go to 149	go to 150	go to 151
1010xxxx11	go to 160	go to 161	go to 162	go to 163	go to 164	go to 165	go to 166	go to 167
1011xxxx11	go to 176	go to 177	go to 178	go to 179	go to 180	go to 181	go to 182	go to 183
1100xxxx11	go to 192	go to 193	go to 194	go to 195	go to 196	go to 197	go to 198	go to 199
1101xxxx11	go to 208	go to 209	go to 210	go to 211	go to 212	go to 213	go to 214	go to 215
1110xxxx11	go to 224	go to 225	go to 226	go to 227	go to 228	go to 229	go to 230	go to 231
1111xxxx11	go to 240	go to 241	go to 242	go to 243	go to 244	go to 245	go to 246	go to 247

	xxxx100011	xxxx100111	xxxx101011	xxxx101111	xxxx110011	xxxx110111	xxxx111011	xxxx111111
0000xxxx11	go to 8	go to 9	go to 10	go to 11	go to 12	go to 13	go to 14	go to 15
0001xxxx11	go to 24	go to 25	go to 26	go to 27	go to 28	go to 29	go to 30	go to 31
0010xxxx11	go to 40	go to 41	go to 42	go to 43	go to 44	go to 45	go to 46	go to 47
0011xxxx11	go to 56	go to 57	go to 58	go to 59	go to 60	go to 61	go to 62	go to 63
0100xxxx11	go to 72	go to 73	go to 74	go to 75	go to 76	go to 77	go to 78	go to 79
0101xxxx11	go to 88	go to 89	go to 90	go to 91	go to 92	go to 93	go to 94	go to 95
0110xxxx11	go to 104	go to 105	go to 106	go to 107	go to 108	go to 109	go to 110	go to 111
0111xxxx11	go to 120	go to 121	go to 122	go to 123	go to 124	go to 125	go to 126	go to 127
1000xxxx11	go to 136	go to 137	go to 138	go to 139	go to 140	go to 141	go to 142	go to 143
1001xxxx11	go to 152	go to 153	go to 154	go to 155	go to 156	go to 157	go to 158	go to 159
1010xxxx11	go to 168	go to 169	go to 170	go to 171	go to 172	go to 173	go to 174	go to 175
1011xxxx11	go to 184	go to 185	go to 186	go to 187	go to 188	go to 189	go to 190	go to 191
1100xxxx11	go to 200	go to 201	go to 202	go to 203	go to 204	go to 205	go to 206	go to 207
1101xxxx11	go to 216	go to 217	go to 218	go to 219	go to 220	go to 221	go to 222	go to 223
1110xxxx11	go to 232	go to 233	go to 234	go to 235	go to 236	go to 237	go to 238	go to 239
1111xxxx11	go to 248	go to 249	go to 250	go to 251	go to 252	go to 253	go to 254	go to 255

1
then" branching

2

3 L00000: 11 -> L00000 go to 0 ; opcode is 3

4

5 L00001: 11.....1. a - b -> a[p]

6 L00002: 11 -> L00000 if no carry go to 0 ; opcode is also 3

7

8 L00003: 1.1.. if s0 = 0

```

 9  L00004:  ....11  -> L00000          then go to 0          ; opcode is still 3
10
11                                     ; the xxxx010100 group can also be programmed as
"if s0 # 1" etc.
12
13  L00005:  ....1.1..          if s0 = 0          ; opcode is 20
14
15  L00006:  ....1.1..          if s0 # 1          ; opcode is 20
0 errors, 0 warnings
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-97 Printer Gear Replacement

Posted by [Katie Wasserman](#) on 15 Nov 2000, 2:07 a.m.

A common problem that causes the premature death of the printer in the HP-97, is the deterioration of the off-white colored spur gear in the drive train. This can be replaced by using the right parts and a little effort.

If your printer is not working, you can easily determine if it's because this gear has deteriorated:

- Remove the 6 recessed screws on the bottom of the calculator and lift off the bottom cover.
- The bottom of the printer assembly is immediately visible, locate the motor and gear train.
- Inspect the middle of the three gears. The pinion gear at the end of the motor shaft is metal and drives the plastic spur gear, which in turns drives another plastic gear that moves the carriage back and forth.
- If your printer problem is due to a damaged middle spur gear, you should be able to see some missing or damaged teeth on the gear.

If this is your problem, you'll need to replace it with a gear that ideally matches all of the following specs:

- 80 pitch - measured by the pitch diameter method (not the circumferential pitch method, my first mistake!)
- 30 teeth
- 1/10" gear face
- 1/4" hub diameter
- 5/64" hub projection
- 3/32" bore
- made of Delrin or nylon

I have not been able find this as a stock item with any manufacturer and I've had no luck with getting this custom made. But I did find a gear that can be made to fit with just a little bit of work. The gear is available from PIC Design as a stock item part number G46-30 for about \$7.00 in single quantities (gears are very expensive items!). Its specs are:

- 80 pitch
- 30 teeth
- 1/8" gear face
- 3/8" hub diameter
- 3/16" hub projection
- 1/8" bore
- made of aluminum

These specs don't look too close on the surface of it, but the things that really count (pitch and teeth) are the same and the other stuff is easy to get around. Here's how.

- (1) Cut off the hub entirely so that it's just a flat gear with no hub projection. Since the gear is aluminum, it's easy to do this with a fine blade hacksaw or a course blade jeweler's saw.
- (2) Adapt the bore from 1/8" to 3/32" by inserting a nylon screw insulator into the hole in the gear. In particular you

should buy part number U-IN-2/2 from Small Parts, Inc.. It's a size #2 x 1/8" length insulator that has exactly the wall thickness that we need. A package of 10 of these sells for \$1.23 (much more reasonable!).

To finish the gear replacement:

- (1) Loosen the 3 screws holding the printer assembly to the plastic frame so that you can lift up the printer the 1/2" or so needed to access the end of the gear shaft.
- (2) Remove the "C" external retaining ring from the end of the shaft and set it aside. If you lose this part (it's very small) you can buy a replacement from Small Parts, Inc as part number U-RSE-1-1/2. A package of 10 sells for \$0.74, so you might as well buy them when you buy the screw insulators. Note, these are functionally the same but are "O" rings not "C". However, they are an exact fit and work just as well.
- (3) Pull the old gear straight off.
- (4) Slip the new gear with the screw insulator inside of it onto the shaft keeping the flange on the end of the screw insulator facing out.
- (5) Put the retaining ring back on. The "O" ring takes a bit of force to push it on, the "C" ring goes on easily.
- (6) Rotate the newly installed gear back and forth with your finger. It should take some force to do this but not a huge amount. If it doesn't move or requires a lot of force remove the gear and try a different screw insulator from the same pack of 10 that you bought, they seem to vary slightly in wall thickness.
- (7) Put the back cover on and tighten the screws.
- (8) Test it!

Here's where to find the needed parts:

Small Parts, Inc.

Address: P.O. Box 4650 - Miami Lakes, FL 33014

Voice: (800) 220-4242

Fax: (800) 423-9009

Web: www.smallparts.com

PIC Design

Address: 86 Benson Road - P.O. Box 1004 - Middlebury, CT 06762

Voice: (800) 243-6125

Fax: (203) 758-8271

Web: www.pic-design.com

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-67 automatic documenting program for Excel

Posted by [John H. Meyer](#) on 10 Sept 2000, 4:54 p.m.

This is an HP-67 automatic documenting spreadsheet which you may [download](#) and run in Excel. It has macros in it so depending on how you have Excel configured you may get a warning about opening spreadsheets that contain macros.

The idea of the spreadsheet is that you can go to the first step of a program and merely type the codes that you see in the display (no spaces needed between pairs of numbers). You press enter, and the program automatically generates a listing. When you are finished, you can add comments in the columns to the right of the listing. Even programs that take up the entire memory take only a few minutes to document.

It would be trivial to adapt the program to work with other HP calculators by simply changing the code in the "Code_Table" worksheet.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP9114 repairs (not for the faint hearted)

Posted by [Steve \(Australia\)](#) on 9 Sept 2000, 12:40 p.m.

9114 Repair Hints

This was a response to a question about problems with an HP9114 disc drive. Maybe this could be useful to someone sometime...

Oh, and by the way, the originally reported fault was that the HP41 reported TRANSMIT ERR when tryind to get the HP9114 to do anything, so read this with that in mind.

But first a message from our sponsor

If you're not familiar with the 9114, then you may be interested in some of the commands that you can use to make it do the basic sort of stuff.

To load a program: READP (filename in alpha)

To save a program: WRTP (note program name in alpha, optionally followed by a comma then the filename (defaults to program name))

To list contents of disk: DIR

To format a new disk: NEWM (prompts for directory size I use 50 when I'm testing)

To delete a program: PURGE (filename in alpha)

Also READA and WRTA to read and write all, READK and WRTK to read and write key assignments, and many others :-)

If you have an HP-IL interface check its quick reference or manual

Back to the problem

I'm assuming that there's no warning lights...

Then (in order of decending probabilities):

- 1) you have a fault in the HP-IL interface of the drive
- 2) the battery level is way too low for the drive to work
- 3) the drive is not turned on

- 4) one of your HP-IL cables just broke
- 5) the HP-IL connectors on the drive are somehow faulty
- 6) you just developed a fault in your HP-IL interface
- 7) you are misreading 0.0000 as TRANSMIT ERR :-)

Assuming that you know how to turn on the unit, then 1 is the most likely, 2 is a possibility, 3 means it's time to slap yourself around a bit, 4 is just silly (assuming you've tested your HP-IL interface on the cassette drive), 5 is a rather remote possibility (I'd put it behind a cable break), 6 is well, i've never actually seen a broken HP-IL interface, and 7 is... umm... a joke :-)

You've got to understand that it's early in the morning here, and I say some silly things when I've just gotten up.

Back to sensibility...

Are you familiar with the start up self test these drives do?

A discussion of the self test

For this unit, turning it on will result in 2 red lights (power and fault) for a few seconds, then the fault light goes out and all is well. If you start it with a disk in it, then the drive does a read/write(?) test on the disk and the whole test takes a bit longer. Likewise, the fault light goes out, leaving the power light on.

Note that the newer 9114B has more lights, and since I don't have mine here to look at, I can't confirm the exact sequence for it.

General fault-finding stuff

Do you have the unit with 2 lights, or 4 (or was that 5)?

Have you replaced the battery? (Or confirmed it it OK) Note that this unit WILL NOT run from the charger alone.

If you turn it on with a disk in it, the drive will do a self test. If this results in the fault light going out, then the drive thinks it's OK.

The drive with 2 lights has no low battery indicator (can anyone confirm this?). The newer version with more lights has them set up as a fuel gauge, i.e. more lights means more charge available. For testing, make sure that they're ALL on, otherwise the drive may decide to stop doing certain things (i.e. writing to the disk).

The internal battery is a SLA battery. It's easy to replace, but if there's problems with your drive, then maybe you don't want to go for that expense. If you're a bit handy with things, then you probably could replace it (temporarily) with 4 nicads of whatever size will fit. Don't expect the charger in the battery pack to work correctly with them though.

Opening up the unit requires a #8 TORX screwdriver. They're the ones that look like an allen key except they have scalloped faces (concave). The screws are so far down little thin holes that you really need a screwdriver. I took my drive under my arm to an electronics store and tried all the TORX drivers until I found one that was the right size and long and thin enough.

Once you take the bottom off (remove the battery first) you'll see the drive and a single large PCB. You'll also see the HPIL connectors and where they connect into the board. Maybe the connectors have become loose or something (I can't see that's likely unless someone's been there before you).

Even if the drive does not work, then there are parts that are valuable in getting another drive to work. At the very least the drive itself (the drives are different between the A and B versions) and the battery pack (some seem to be missing this) are items that you can swap to other units.

The drives are so big and chunky that you can actually repair them! I removed and reattached the head of one (in a 9114A) because it had become bent out of shape (well the suspension for the head had). If you think it's fiddly to put a card reader back together, then a disk drive is even worse. There's springs in there that work against you.

Oh, check that your drive has heads! The common fault of these drives is that the lubricant hardens and the eject mechanism slows down. After a point, the eject must be completed manually (i.e. you have to pull the disk out) If this is the case, then it is quite possible that the heads are not out of the way, and the disk shutter will tear them off (or in my case just bend them way out of shape). If pushing the eject button (without a disk in the drive) just makes something in the drive go up and down, and it doesn't latch, then your drive has it BAD and may have head damage.

Fixing eject problems

(Check out the next section for a blow by blow description of opening the drive up)

This too is easy to fix if you catch it early. Take out the drive (the earth lead can be a pain to remove, and you will probably have to move the the main board (just a little) to get the screws back in (same as the ones that hold the cover on). Then take off the cover off the 3 1/2 inch drive. 1 screw does this on both types I think?? Or maybe one clips off. Then you'll be able to see the levers and things that make the eject mechanism work. They are all down one side of the drive.

Now get a cotton swab or something and using alcohol, or degreaser or something that will act as a solvent for the grease, wipe away the old hard grease. remember that it's behind things too, so work the eject mechanism, and get the grease out from behind everything you can. At some point the eject mechanism will start to work again. If it hasn't been latching, the first indication is when a small lever jumps up and locks the drive in the eject position. This is a good thing, but it does not mean you've got all the gunk out. The eject should be snappy. When it is, put the drive down and come back tomorrow.

The drive will no longer work! The solvent had softened some of the grease, and heat had done it's part too. So start again, and get the drive working again. (Then again, maybe you're lucky (or thorough) and did a good job the first time)

Keep repeating this until the drive works the next day, or you get sick of it.

Now use a light machine oil to LIGHTLY lubricate the parts that obviously move against one another and work the drive to make sure the oil gets to where it's needed.

Then put it all back together.

OK, I'll admit it. After I get the thing ejecting again I put it back together to make sure everything still works, but sure as eggs, it's gummed up again the next day, so I go back through the routine again until I've done it properly.

I've used this technique on about 10 drives of this vintage, and assuming there was no head damage, have had 100% success. I've only had one drive with head damage that was recoverable. In hindsight, the only reason why I attempted the fix this one, rather than earlier ones is that when I first did this there were quite a lot of drives (the 3 1/2 drives, not the 9114 units) around, and binning one was not such a bad thing.

Repairing a damaged head

Note that this applies to the drive in the 9114A. The drive in the 9114B is different and does not seem to suffer from the lubricant problem (so far) and thus does not have the broken head problem.

First open the 9114. Then remove the drive (the big silver thing DUH!). Use the #8 TORX driver to remove the 4 screws holding the aluminium plate the drive is on. Remove the two connectors, and the earthing strap.

Then you'll find the plate is held on by 4 philips screws (why not TORX?) Don't bother to remove this yet.

There's one funny screw at the back of the drive, but a normal screwdriver will remove it. Take this off, then the shielding from the drive.

Oh, and check out the model Number. It's a SONY Model OA-D43W-11. For some real fun, try locating a new one of these :-)

If you look at the side opposite the serial number then you'll see a number of levers and springs that do an awful lot to eject the disk. If you're reading this section before cleaning and lubricating the drive, then this is what you have to clean.

If you're going to repair the head, then clean this first, or you'll be repairing the head again real soon...

In any case, insert a disk and eject it a few times to see how it works.

In brief, head is up. disk slides in, and shutter is opened. disk goes in a little further and a lever is moved into a slot allowing the heads to drop and preventing the disk from being forced out. The movement of this lever also allows another slider to move back, dropping the disk into the drive.

Pressing eject does the reverse. eject lever pushed slider which lifts disk. lever drops into slot lifting heads and freeing the other slider which forces the disk out. On the way out the shutter closes.

With the gunky lube problem the movement of the lever that lifts and lowers the heads is inhibited. It tends to stay in the head down position. Thus the disk can't eject, and when the user pulled the disk out by hand, the head catches on the front of the disk and bends it (if you're lucky) or tears it off (if you're not).

Here is a procedure to fix a bent head.

First fix the gunky lube problem (did I mention this?)

Then check to make sure that the head isn't really OK. It may rest at a bit of an angle, but as long as it's flat on the disk when it's down then that's OK.

Assuming it's not OK

The first (and easiest) thing to do is to fix the head.

You can see that there are 4 small phillips screws holding the head mechanism onto the drive. If you look carefully you'll also notice a spring under and between the two closest to the front of the drive. Take a long hard look at that spring and the way it's connected to things. As soon as you remove those 2 screws that spring will snap back inside the drive and you'll have a really hard time getting it back.

You should also make some notes as to where the head sits, because you're going to really stuff up the head alignment too!

OK, remove the base plate (you'll need to get at that side of the drive to get the bits that are about to fall out :-)

Now remove the 2 front screws. I'm not sure whether it's better to do this with a disk in or not. I did it with the disk out. BANG! spring snaps back, small piece of metal falls out, head is loose.

Now remove the back pair of screws. This is not so dramatic.

Now you can carefully lift the head back and turn it over. Then, if nothing is broken you can try to prod it back into shape. Be really careful. There's some fine wires and the head suspension is very delicate.

That's the easy part!

Now the moderately difficult part.

Fold the head assembly back and figure out how to place the back screws on again. Fiddly, but not hard.

Then figure out how to get the front 2 screws back. It's wuate entertaining, but I won't spoil it by telling you how many times you will utter rude words while you try.

Then the hard bit. You've got to get that bloody spring back on! OK, this was the hardest bit, but this is a good way to do it.

- 1) get some dental floss (this is really strong, you could try cotton, but then I would recommend floss, because you really should floss!).
- 2) hold both ends of the floss to form a loop, hang the spring on the loop. Now loop a few turns around the spring so it's less likely to fall off. DONT tie it on.
- 3) turn the drive over and lower the spring into the drive through a hole just next to where the spring has to go.
- 4) somehow snag the spring onto the hook on the head assembly.
- 5) keeping tension on the spring with the floss, stretch the spring back and hook it onto the other hook.
- 6) let go of one end of the floss and carefully pull it out.

You may like to losten the screws slightly and try to put the head into similar alignment to what it had before. (good luck). Actually if you have a disk recorded on another drive (or one recorded previously on this one) then you stand some chance of getting passabe alignment by trial and error, but you parience may need to be infinite!

Put the cover back on, and the base place, replace the drive and test it.

With any luck it now works again!

Mine did!

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

IR Printer information

Posted by [Steve \(Australia\)](#) on 31 July 2000, 11:00 a.m.

Maybe someone else will find this of interest. I know I was looking for this for a long time.

Most of the correct stuff ;-) comes from the HP Journal article October 1997 pp16-20. The rest is based on my observations in writing code to decode the output from my HP41 IR Printer interface.

Basics

The carrier frequency is 32.768Khz. (for the HP-18C)

The carrier has an even mark and space of approximately 15.26uS

The carrier is modulated by a square wave with a fixed mark, but variable space. Thus the information is carried by the gaps between pulses, **not the pulses themselves!**

Data Encoding

The data is encoded bit by bit. Bit times are fixed (28 pulses of the carrier, or approx 854.5uS). This bit time is broken into two halves.

- * A 1 bit is defined as a burst in the first half of the bit time, and no pulse in the second half
- * A 0 bit is defined as a burst in the second half, but not in the first half.
- * A start bit is defined as a burst at the beginning of three successive half bit times (an otherwise illegal combination)

Note that this makes a start bit 50% longer than a normal bit.

An Example

If I were drawing things, it would look like this:

(this is a start bit followed by a 1 and a 0.) Note that you don't start counting until you see the first rising edge. To make things easier, I only show 4 pulses per 1/4 bit time, rather than the 6 to 8 that you may see in practice.

Also note that while HP talks of 1/2 bit times, you can usefully think in terms of 1/4 bit times with the even 1/4's ALWAYS in a space, and the mark only ever occurring in one of the odd 1/4's.

___|_|_|_|___|_|_|_|___|_|_|_|___|_|_|_|_____||_|_|_|___

To break that up:

is some amount of space between characters

||||____||||____||||____

is three half bit times with pulses in their first half. (and is thus a start bit)

||||_____

is two half bit times with pulses in the first half bit time (a 1)

_____||||____

is two half bit times with pulses in the second half bit time (a 0)

Data is sent in frames. Each frame consists of a start bit, 4 ECC bits, and 8 data bits.

If you look at the output of one of these IR devices on a storage scope, or via some other method, you'll notice immediatly that the frames stand out like islands of data in the sea of noise. As you look closer you'll see the groups of pulses and the spacing between them. You'll notice the three different size gaps. These are 1 unit, 3 units, and 5 units. Coding these gaps is actually one way of coding the data.

A Table to Cheat By :-)

Here is a table that some of my software uses to decode these gaps. Note that it was devised BEFORE I knew anything about how the data was decoded.

```
const
  FrameChars : Array[0..255] of string[14] = (
    '11222222222222', // 0
    '11221232222221', // 1
    '11213132222213', // 2
    '11212322222212', // 3
    '112123222222132', // 4
    '112131322222131', // 5
    '112212322222123', // 6
    '112222222222122', // 7
    '1113213222221322', // 8
    '1113132222221321', // 9
    '1112322222221313', // 10
    '1112223222221312', // 11
    '1112223222221232', // 12
    '1112322222221231', // 13
    '1113132222221223', // 14
    '1113213222221222', // 15
    '11131322222213222', // 16
    '11132132222213221', // 17
    '11122232222213213', // 18
    '11123222222213212', // 19
    '11123222222213132', // 20
    '11122232222213131', // 21
    '11132132222213123', // 22
    '11131322222213122', // 23
    '11221232222212322', // 24
    '11222222222212321', // 25
    '11212322222212313', // 26
    '11213132222212312', // 27
    '11213132222212232', // 28
    '11212322222212231', // 29
    '11222222222212223', // 30
    '11221232222212222', // 31
    '111232222222132222', // 32
    '111222322222132221', // 33
    '111321322222132213', // 34
    '111313222222132212', // 35
    '111313222222132132', // 36
    '111321322222132131', // 37
    '111222322222132123', // 38
    '111232222222132122', // 39
```

'11213132131322'	//	40
'11212322131321'	//	41
'11222222131313'	//	42
'11221232131312'	//	43
'11221232131232'	//	44
'11222222131231'	//	45
'11212322131223'	//	46
'11213132131222'	//	47
'11212322123222'	//	48
'11213132123221'	//	49
'11221232123213'	//	50
'11222222123212'	//	51
'11222222123132'	//	52
'11221232123131'	//	53
'11213132123123'	//	54
'11212322123122'	//	55
'11122232122322'	//	56
'11123222122321'	//	57
'11131322122313'	//	58
'11132132122312'	//	59
'11132132122232'	//	60
'11131322122231'	//	61
'11123222122223'	//	62
'11122232122222'	//	63
'11122321322222'	//	64
'11123131322221'	//	65
'11131231322213'	//	66
'11132221322212'	//	67
'11132221322132'	//	68
'11131231322131'	//	69
'11123131322123'	//	70
'11122321322122'	//	71
'11212231321322'	//	72
'11213221321321'	//	73
'11221321321313'	//	74
'11222131321312'	//	75
'11222131321232'	//	76
'11221321321231'	//	77
'11213221321223'	//	78
'11212231321222'	//	79
'11213221313222'	//	80
'11212231313221'	//	81
'11222131313213'	//	82
'11221321313212'	//	83
'11221321313132'	//	84
'11222131313131'	//	85
'11212231313123'	//	86
'11213221313122'	//	87
'11123131312322'	//	88
'11122321312321'	//	89
'11132221312313'	//	90
'11131231312312'	//	91
'11131231312232'	//	92
'11132221312231'	//	93
'11122321312223'	//	94
'11123131312222'	//	95
'11221321232222'	//	96
'11222131232221'	//	97
'11212231232213'	//	98
'11213221232212'	//	99
'11213221232132'	//	100
'11212231232131'	//	101
'11222131232123'	//	102
'11221321232122'	//	103
'11131231231322'	//	104
'11132221231321'	//	105
'11122321231313'	//	106
'11123131231312'	//	107
'11123131231232'	//	108
'11122321231231'	//	109
'11132221231223'	//	110
'11131231231222'	//	111
'11132221223222'	//	112
'11131231223221'	//	113
'11123131223213'	//	114
'11122321223212'	//	115
'11122321223132'	//	116
'11123131223131'	//	117
'11131231223123'	//	118
'11132221223122'	//	119
'11222131222322'	//	120

'11221321222321'	//	121
'11213221222313'	//	122
'11212231222312'	//	123
'11212231222232'	//	124
'11213221222231'	//	125
'11221321222223'	//	126
'11222131222222'	//	127
'11212223222222'	//	128
'11213213222221'		
'11221313222213'		
'11222123222212'		
'11222123222132'		
'11221313222131'		
'11213213222123'		
'11212223222122'		
'11122313221322'		
'11123123221321'		
'11131223221313'		
'11132213221312'		
'11132213221232'		
'11131223221231'		
'11123123221223'		
'11122313221222'		
'11123123213222'		
'11122313213221'		
'11132213213213'		
'11131223213212'		
'11131223213132'		
'11132213213131'		
'11122313213123'		
'11123123213122'		
'11213213212322'		
'11212223212321'		
'11222123212313'		
'11221313212312'		
'11221313212232'		
'11222123212231'		
'11212223212223'		
'11213213212222'		
'11131223132222'		
'11132213132221'		
'11122313132213'		
'11123123132212'		
'11123123132132'		
'11122313132131'		
'11132213132123'		
'11131223132122'		
'11221313131322'		
'11222123131321'		
'11212223131313'		
'11213213131312'		
'11213213131232'		
'11212223131231'		
'11222123131223'		
'11221313131222'		
'11222123123222'		
'11221313123221'		
'11213213123213'		
'11212223123212'		
'11212223123132'		
'11213213123131'		
'11221313123123'		
'11222123123122'		
'11132213122322'		
'11131223122321'		
'11123123122313'		
'11122313122312'		
'11122313122232'		
'11123123122231'		
'11131223122223'		
'11132213122222'	//	191
'11132122232222'	//	192
'11131312322221'		
'11123212322213'		
'1112222322212'		
'11122223222132'		
'11123212322131'		
'11131312322123'		
'11132122322122'		
'11222212321322'		
'11221222321321'		

'11213122321313'		
'11212312321312'		
'11212312321232'		
'11213122321231'		
'11221222321223'		
'11222212321222'		
'11221222313222'		
'11222212313221'		
'11212312313213'	//	210
'11213122313212'		
'11213122313132'		
'11212312313131'		
'11222212313123'		
'11221222313122'		
'11131312312322'		
'11132122312321'		
'1112222312313'		
'11123212312312'		
'11123212312232'	//	220
'1112222312231'		
'11132122312223'		
'11131312312222'		
'11213122232222'		
'11212312232221'		
'11222212232213'		
'11221222322212'		
'1122122232132'		
'11222212232131'		
'11212312232123'	//	230
'11213122232122'		
'11123212231322'		
'1112222231321'		
'11132122231313'		
'11131312231312'		
'11131312231232'		
'11132122231231'		
'1112222231223'		
'11123212231222'		
'111222223222'	//	240
'11123212223221'		
'11131312223213'		
'1113212223212'		
'1113212223132'		
'11131312223131'		
'11123212223123'		
'111222223122'		
'11212312222322'		
'11213122222321'		
'11221222222313'	//	250
'11222212222312'		
'11222212222232'		
'11221222222231'		
'11213122222223'		
'11212312222222'	) ; //	255

This table encodes the GAPS between modulation pulses for each character. 1 indicates a short gap, 2 the middle size gap, and 3 the long gap. Knowing that the pulses fit around each gap, that the gaps are in the proportion 1:3:5, and that you should send 7 carrier pulses at 32767Hz, this is probably enough info.

How To Cheat

So, character A is 65 and is '11123131322221'

You would send:

|_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|

Where each | is 7 carrier pulses, and each _ is a gap of equivalent size.

If you're further interested...

[illegible]

And yes the 1's do extend out beyond the end! This is because the last set of pulses occur in the first quarter of the last bit time (first half of the first half of the last bit time!!!). So even though we're encoding information in the gaps, the last gap does not need to be terminated.

SSSSSS = start bit

1111 = 1 bit

0000 = 0 bit

So it was S110101000001

1101 is the ECC 01000001 is the code for A in ASCII MSB first

Calculating the ECC

How do we get the ECC? well it's like this...

The 4 bits (call them H1, H2, H3, and H4) are calculated individually

Firstly for each bit you apply a mask (AND) to the data

the masks are:

H1 01111000

H2 11100110

H3 11010101

H4 10001011

So for the code 01000001 the resulting data for these is

H1 01000000

H2 01000000

H3 01000001

H4 00000001

Then you calculate the EVEN parity bit required for these (i.e. the bit required so that there are an EVEN number of bits. Since H1, H2, and H4 have an odd number of bits, the parity bit for these would be 1, bit for H3 it is 0.

Then you string them together (the parity bits) this gives you:

1101 which is what we found above!

These parity bits can be used to detect and correct up to two missed bits (though how you do that is well beyond what I'm prepared to go into!)

But, my software DOES decode these without resorting to calculating the ECC bits (although I may add that some time)

But you're trying to SEND this data right? So the lookup table I've given you should be all you need, unless you want to do all the hard work yourself :-)

P.S. (Timing made easy)

An easy way to play with this is to capture it with your sound card... (Oh damn, I've let out my secret). It has the

advantage of handling all the timing for you, so software speed is no longer an issue. This is **especially** important when using an operating system that multi-tasks.

P.P.S. (Missing bits on received data)

This table indicates what you should do when you receive a bit (i.e. a rising edge) after x timer ticks (a timer tick is 1/32768 sec)

Ticks	Bit	Comment
0 - 2		Ignore (too short)
3 - 7	X	2 pulses in the same bit time (treat as a missed bit)
8 - 12	1	
13 - 17	0	
18 - 23	X 1	Missed bit followed by a 1
23 - 28	X 0	Missed bit followed by a 0
28 - 34	X X 1	2 missed bits then a 1
33 - 39	X X 0	2 missed bits then a 0

You can use the ECC to determine what the bits are, as long as you know which are missing (which you probably will).

What I do is to decode the gaps, then use some heuristics to determine what the gap is likely to be, then do pattern matching to determine which is the most likely. This is probably too code intensive for a PIC, but the calculation of bits using the ECC should be OK.

P.P.P.S. (Printer Buffer Management)

Because the IR interface is one-way, there is no way to detect that the buffer is full. Thus the sender needs to send data in such a way that the buffer never overflows.

With new batteries, the IR printer can print about 2 lines per second. When the batteries are almost exhausted, this drops as far as 1 second per line.

The buffer in the printer can contain 200 characters.

The HP way is to have a rather simple algorithm that assumes each line has 25 bytes. Thus it assumes the buffer will hold 8 lines. The calculator simply keeps track of how many lines have been sent, and how long it has been since the last line was sent. Before a line is sent, the calculator determines how much time has elapsed since the last line was sent, and decrements the assumed number of lines in the buffer by the appropriate amount. If the number is less than 8, it allows a line to be printed, otherwise it waits.

P.P.P.P.S. (Error Correction)

I know I promised I wouldn't go into this, but I lied! (but I won't go into this in great detail)

If you have missing bits (up to two of them) in the data area of the frame, the ECC can be used to determine what they are.

For each bit in the data, there are 2 (or 3) parity bits that have been calculated using this bit. If there is only a single missing bit, then any of these can be used to determine what the missing bit is.

For example, suppose the bits received have been:

11010100x001

The X represents a missing bit. From looking at the masks for the ECC bits, we see that H1 and H4 have been calculated using this bit.

so

```

      0100x001  (data)
AND  01111000  (H1 mask)
-----
      0100x000  with EVEN parity of 1  (from first bit of 1101)

```

If we add all the 1s up (in the result and the parity) we have 2 which is even, therefore the missing bit must be a 0

Lets assume a more difficult case:

11010x00x001

There are 2 missing bits, but you can see that only the first missing bit is in H3, and only the second is in H4, so we can calculate them individually:

```

      0x00x001  (data)
AND  11010101  (H3 mask)
-----
      0x000001  with EVEN parity of 0  (from first bit of 1101)

```

Note that we only get an x in the result if the mask has a 1 in that position.

The 1s have to add up to an even number (always) and in this case there's an odd number, so this missing bit is a 1.

The other missing bit is calculated in a similar manner.

Missing bits can also happen in the ECC area too. If that's the only place they are, then there's really nothing to worry about. If there's one in this area, and one in the data area, then the remaining ECC bits will provide enough information to correct the single bit in the data area.

All of this is based on the fact that we can determine that a bit is actually missing. This is a fair assumption, because the bits are far more likely to be missed than mistakenly detected as a 1 rather than a 0 (or vice versa). However it can happen.

Assume you get this:

Note that you can easily correct for 2 missing bits, and sometimes 3. But it's not always this easy...

110101001001

This all seems OK, but what happens if we recalculate the ECC?

H1 calculates as 0

H2 calculates as 1

H3 calculates as 0

H4 calculates as 0

So we got an ECC of 1101 but calculated 0100.

What we do is XOR the received and calculated ECC

```
XOR  1101  {received}
     0100  {calculated}
     1001
```

Now we AND the masks of H1 and H4 to find out what bit is incorrect

```
AND  01111000  {H1}
     10001011  {H4}
     00001000
```

And this tells us which bit has been flipped.

But this is really a special case. Consider this bit pattern

110101100001

H1 calculates as 0

H2 calculates as 0

H3 calculates as 0

H4 calculates as 1

So we got an ECC of 1101 but calculated 0001.

So we do is XOR the received and calculated ECC

```
XOR  1101  {received}
     0001  {calculated}
     1100
```

Now we AND the masks of H1 and H2 to find out what bit is incorrect

```
AND  01111000  {H1}
     11100110  {H2}
     01100000
```

Oh dear!!!!!! 2 bits!!! Well not really. To fully determine the bit we need to also OR together the other masks

```
OR   11010101  {H3}
     10001011  {H4}
     11011111
```

Then invert this

00100000

then AND this with the result above:

```
AND  01100000  {H1 AND H2}
     00100000  {NOT (H3 OR H4)}
     00100000
```

Which singles out the bit. What we've done is to find out what is common in the masks of the ECC that were calculated in error and removed ALL the bits of the ECC that calculated correctly. **Note that this also applies to the missing bit problem too, but if you didn't read this far, you're not really serious :-)**

All a bit anal really :-) But useful if you're trying to recover data out of noise.

You may ask, what happens if a bit in the ECC is flipped? The answer is that only a **single** ECC bit will calculate in error, and this means that either multiple bits in the data are in error, or that the ECC was corrupt.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Some information on the CCD-Rom -W&W CCD A or -W&W CCD B

Posted by [Reinhard Breuer](#) on 24 July 2000, 11:10 p.m.

Some information on the CCD-Rom -W&W CCD A or -W&W CCD B.

It was built by the german company W&W Software. This company currently exists, but they don't give any response on questions about this module.

This module extends all models of HP41 but best is a CX or having X-Functions- and Time-Module plus printer and IL.

This description is made from the quick-reference-card.

It uses XROM-numbers 09,00 to 09,63 and 11,00 to 11,41.

Cat 2 Headlines:

XROM	Title	contains functions
09,00	-W&W CCD A or -W&W CCD B	--> General functions
09,08	-ARR FNS	--> Array functions (Arrays could be defined
	in Data-Registers or in X-Memory-Files)	
09,47	-HEX FNS	--> Hex functions
11,00	-I/O FNS	--> I/O functions (provides a special function)
11,17	-ADV FNS	--> ADV-Functions (provides a special function)
11,35	-XF/M FNS	--> XF-Memory-Functions (provides a special function)

Incompatibilities:

If used together with the IL-Development-Module, the CCD-Module must be plugged into a lower portnumber than the IL-Development-Module, otherwise the OS-extensions will not work.

If used together with a ZENROM, the CCD-lowercase-mode should be disabled (register 13 byte 4 bit 7 has to be set to 1).

The OS-extensions will not work on HP41 with serialnumbers less than 2035! There is a workaround available (see special headline-functions).

Special headline-functions:

-XF/M FNS

This function either prompts for a 4 digit Hexadecimal-ROM-address or uses a 2 character string in ALPHA treated as

a ROM-address ("@A" -> \$4041), which will be called. With this function it is possible to jump to any microcode instruction in the whole 64K address-space. However, only in bank 0!

Workaround for ASN, CAT, XEQ:

The call addresses are as follows (PORTx is the slot-number where the module is plugged in):

	PORT1	PORT2	PORT3	PORT4
ASN	: 9869	B869	D869	F869
CAT	: 984A	B84A	D84A	F84A
XEQ	: 9865	B865	D865	F865

Manual handling:

Assign -XF/M FNS (XROM 11,35) to a key. Press this key. Enter the above value of the preferred function. The function will be executed.

Programmatical handling:

create a textline containing two characters, where the first is the upper half of the address and the second is the lower half of the address. Press the assigned key (see manual handling)

```
01 'XX < character 98 (hex) + character 69 (hex)
02 -XF/M FNS
```

this lines will do the same as described under manual handling.

Several synthetik way exist for entering program bytes to the memory.

-I/O FNS

```
This function checks if a printer is connected.
Printer Mode      flag 25  action
no      running a program clear  STOP-execution
no      running a program set   CF 25, continue execution
no      manual      clear    message 'NO PRINTER'
no      manual      set      CF 25
yes     running a program clear \
yes     running a program set   / continue execution (like NOP)
yes     manual      clear    \
yes     manual      set      / nothing (like NOP)
```

-ADV FNS

This function prompts 'KEY?' and waits for a single key. The 'ON'-key escapes, any other key sounds a short tone and puts it's keycode to X.

Statusbits used (only ??) by the CCD-module:

register byte bit usage

13	4	7	
		1	disabled CCD-lowercase-mode
		0	enabled
13	4	6	
		1	on CCD-autostart
		0	off
13	4	5	
		1	2. part CCD-input-sequence
		0	1. part
13	4	1 0	
		0 0	unsigned Hexmode
		0 1	2cmp
		1 0	1cmp
		1 1	(not documented)

Operatingsystem extensions:

CAT 0..F general on Cat 0,2,4,6,8-F: SHIFT-state will be active until
SHIFT-key is pressed again

CAT 0 displays ID/AID of IL-peripherals
ENTER selects the device
C forces Device Clear

CAT 1 calls standard catalog 1

CAT 2 shows headings only
ENTER opens the functionblock
XEQ executes or programs that function
A assigns that function

CAT 3 calls standard catalog 3

CAT 4 like EMDIR (X-Functions-Memory-Module), but Buffer-, Matrix- and
Key-Files are shown correctly

CAT 5 calls ALMCAT-function of Time-Module

CAT 6 Shows any Key-Assignments but synthetic codes are shown correctly
'TEXT7' or 'RCL M'
C deletes an assignment

CAT 7 calls DIR-function of IL-module

CAT 8..F like CAT 2, but beginning with the module on addressblock 8..F
CAT 8 port 1 lower ROM
CAT 9 port 1 upper ROM
CAT A port 2 lower ROM
CAT B port 2 upper ROM
CAT C port 3 lower ROM
CAT D port 3 upper ROM
CAT E port 4 lower ROM
CAT F port 5 upper ROM

ASN also allows decimal/hex input of any possible 2 byte sequence
(144 117 for RCL M)

XEQ also allows decimal/hex input of any possible 2 byte sequence
(144 117 for RCL M)

CALC-mode direct input of ANY stack register function just like RCL.Z or
STO IND T, ...
RCL.e STO.c X<>.a STO IND .M RCL IND P ...

ALPHA-mode direct upper-/lower-case input to ALPHA includes also decimal and hex entry of charactercodes!

General functions:

B? exists a buffer with this identity (X) (CCD uses ID 5 for WSIZE,
actual MATRIX, ...)

CAS clear CCD-autostartbit

CLB clear buffer with this identity (X)

RNDM get a random number ($0 \leq N < 1$)

SAS	set CCD-autostartbit
SEED	define a startvalue for RNDM
SORT	sorts a range of registers (Riii, Rjjj) X=iii,jjj or X=jjj,iii up/or down includes ALPHA

Array-functions:

Array functions require an information in ALPHA in the form of OPERAND1,OPERAND2,RESULT.

Arrays can be located in a X-Memory-file (any name is allowed) or in USER-memory (name is Rxxx, xxx is the base register).

>C+	X to column and increment i, column oriented dataentry
>R+	X to row and increment i, row oriented dataentry
?IJ	retrieve the current pointer to X in the form iii,jjj
?IJA	retrieve the current pointer of the array defined in ALPHA to X in the form iii,jjj
C<>C	exchange columns, exchanges columns kkk and lll (kkk, lll in X)
C>+	from column to X and increment, column oriented dataretrival
C>-	from column to X and decrement, column oriented dataretrival
CMAXAB	column maximum absolute, sets the pointer to the largest absolute element of the specified column (in X) and reads it to X
CNRM	column norm, calculates the columnnorm of the named array
CSUM	column sum, calculates all column sums, requires OPERAND1,RESULT in ALPHA
DIM	dimension, retrieves the dimension of the named array (in ALPHA)
FNRM	Froebius norm, calculates the Froebius-norm of the named array (in ALPHA)
IJ=	set arraypointer of current array to iii,jjj (in X)
IJ=A	set arraypointer to ii,jjj (in X) of array (in ALPHA)
M+	matrix +, adds all elements of two matrices, requires OPERAND1,OPERAND2,RESULT in ALPHA
M-	matrix -, subtracts all elements of two matrices, requires OPERAND1,OPERAND2,RESULT in ALPHA
M*	matrix *, multiplies all elements of two matrices, requires OPERAND1,OPERAND2,RESULT in ALPHA
M*M	multiplies two matrices, requires OPERAND1,OPERAND2,RESULT in ALPHA

M/ matrix /, divides all elements of two matrices, requires OPERAND1, OPERAND2, RESULT in ALPHA

MAX maximum element, sets matrixpointer to the largest element of the given array and returns it's value to X

MAXAB maximum absolute element, sets matrixpointer to the largest absolute element of a given array and returns it's value to X

MDIM create a matrix, NAME in ALPHA, size mmm,nnn in X

MOVE moves on part (Aij to Akl) of an array into a second one (Bnm)
 ALPHA: OPERAND1, RESULT
 X : iii, jjj
 Y : kkk, lll
 Z : mmm, nnn

PIVOT move the array pointer to the Pivot-element of the array and returns its value to X

R-PR row - P * row, P in X (LR)

R-QR row - Q * row, Q in X (Gauß)

R<>R exchange rows kkk and lll, kkk, lll in X

R>+ from row to X and increment, row oriented data retrieval

R>- from row to X and decrement, row oriented data retrieval

R>R? compare all elements of row kkk to all elements of row lll and skips the next instruction, if one element of row lll is greater than the corresponding element in row kkk, kkk, lll in X

RMAXAB row maximum absolute,, sets the pointer to the largest absolute element of the specified row (in X) and reads it to X

RNRM row norm, calculates the rownorm of the named array

RSUM row sum, calculates all row sums, requires OPERAND1, RESULT in ALPHA

SUM sum, adds all elements of the matrix, stores result in X

SUMAB sum absolute, adds absolute values of all elements of the matrix, stores result in X

SWAP exchanges two blocks of two arrays (see MOVE)

YC+C Y * column + column, adds column kkk multiplied by Y to column lll, kkk, lll in X

Hexfunctions:

1CMP 1's complement mode

2CMP 2's complement mode

AND X AND Y -> X

bC?	X-bit in Y clear?
bS?	X-bit in Y set?
Cb	clear X bit in Y
NOT	X = NOT X
OR	X OR Y -> X
R<	rotate Y left by X (FLAG 0 IS CARRY)
R>	rotate Y right by X "
S<	shift Y left by X "
S>	shift Y right by X "
Sb	set X bit in Y
UNS	unsigned mode
WSIZE	presets the bitsize for all hex-fns (1..32 bits)
XOR	X XOR Y -> X

I/O functions:

ABSP	alphabackspace (one character)
ACAXY	accumulate ALPHA by X and Y to Printerbuffer ALPHA left, X right, total width in Y
ACLX	accumulate line by X (aa,bbb) puts bbb count of char aa into printerbuffer
ARCLE	ARCL's 10 E3 into ALPHA as 10K 23 E7 -> 230M 2.2 E-9 -> 2.2N ...
ARCHLH	ARCLS's X as HEX-CHARS into ALPHA depends on mode and WSIZE
ARCLI	ARCL's 33.33 into ALPHA as 33
CLA-	deletes ALPHA from right to left until a space is found 'ABCD 123' --> 'ABCD '
F/E	sets FIX/ENG-Mode
INPT	universal input routine prompts ALPHA, allows input from Riii to Rjjj stepby cc includes range-checking R000 iii,jjjcc R001 min-value R002 max-value
PMTA	same as PROMPT, AON, STOP, AOFF sequence

PMTH	prompts ALPHA, asks for a HEX-number (see WSIZE)
PMTK	prompts ALPHA, get only allowed keys, returns it's position 'Text JN' N returns 2, J returns 1 unallowed chars rejected with a tone, one or more spaces required before valid keys!
PRAXY	like ACAXY, but directly printed
PRL	prints a line of '-' chars (depending on flag 12)
VIEWH	shows X as HEX (see WSIZE)
XTOAH	like XTOA, instead of a char a Hex-string is appended

ADV-Functions:

A+	increments RAM-address in X (X=RRR,B) one byte
A+B	address(X) +-bytes(Y)
A-	decrements RAM-address in X (X=RRR,B) one byte
A-A	subtracts address in X from address in Y result is bytes
DCD	decodes X to ALPHA (appends) 14 HEX-digits
PC<>RTN	exchanges PC with the first return-address
PC>X	gets the current PC into (X=RRR,B)
PEEKB	gets one BYTE from RAM like RCL IND X
PEEKR	gets one REGISTER from RAM like RCL IND X
PHD	Program head, returns the address of the first instruction (not programmable)
PLNG	returns the length of a program in bytes
POKEB	STO IND Y see PEEKB
POKER	STO IND Y see PEEKR
PPLNG	programmable PHD
X>PC	sets the PC from X
X>RTN	sets first RTN-address from X
XR>RTN	sets first RTN-address to any line of a program in a ROM, X=linenumber, Y=kk,ll (XROM-number)

XF-Memory-Functions:

GETB	GET Buffer from X-MEM-file
GETK	GET Keys from X-MEM-file (deletes any userkeys before)
MRGK	MeRGes Keys from X-MEM-file
SAVEB	SAVE Buffer to X-MEM-file
SAVEK	SAVE Keys to X-MEM-file
SORTFL	sorts a file (only up)

Hints:

ALPHA lowercase mode may not work correctly in REV. A, when in PRGM-mode.

Programming example:

Title: Change Programs Privateflag (CPP)

Description: Execute CPP. On prompt key in a name of a global label plus R/S. CPP will change this program from being private or vice-versa.

```
01 LBL "CPP"
02 CF 25      ; do not allow errorskipping!!
03 'NAME:    ; ask user for the name
04 PMTA      ; stop here with ALPHA on and wait for input, press R/S to continue
05 PHD       ; retrieve the address of the first instruction of the wanted program
06 PPLNG     ; retrieve the size of that program in bytes
07 1         ; the private flag is stored in the last byte of the program
08 -         ; subtract this from the size to get address of the last byte
09 CHS       ; program-memory goes from upper to lower addreses, so subtract
10 A+B       ; calculate startaddress + distance
11 PEEKB     ; retrieve the last program byte
12 64        ; Bit 6 holds the private-flag
13 XOR       ; toggle private-flag
14 POKEB     ; put the last program byte back
15 CLA       ; clear alpha
16 CLX       ; clear X
17 STO Y     ; clear Y
18 STO L     ; clear Last-X
19 END       ; 43 Bytes used, 15-17 clear only used registers,
              ; if used CLST , line 16 could be skipped, saving two bytes
```

Hope this will help.

For questions mail to: reinhard.breuer@imail.de

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Repairing pre-recorded cards for HP-65, 67, 97

Posted by [Erik Wahlin](#) on 20 July 2000, 8:50 a.m.

I just wanted to share a trick I came up with to fix HP pre-recorded cards from HP program pacs that have had their data erased or damaged. Every now and then, I have a problem reading a card with my HP-65 or HP-67/97. Sometimes it is only one side that is un-readable. This trick may not work if the magnetic strip is physically damaged. The basics: Each corner of the card has a diagonal cut for write protection of that particular side of the card. The trick is to build up the corner with scotch tape so that it engages the write circuitry of the card reader. I begin by applying one strip of tape on one side of the card over the cut corner. I usually lay about 3 mm of tape on the card, and the tape extends over the corner. Then I trim the tape using scissors to form the corner. Next, I build up the corner on the other side by applying at least 4 layers of tape ONLY OVER THE CORNER SECTION. Do not apply any more tape over the magnetic card because it will make the card too thick to get through the card reader. After each application, trim the excess tape to make a nice rigid corner. If the problem on the card is side one, you are in luck because you only have to cover the side 1 corner. If the problem is on side 1 and side 2 or just side 2, then you have to cover both corners. The reason for this will be clear below. On the HP-65 you only have one side to worry about. Next enter as much of the program that you can by running the card through the calculator with the program switch set at "run". You will obviously get "error" trying to run the card through the bad side. Next, enter the remainder of the program by hand. You will need the program listing from the application pac manual. Next, set program switch to "w/prgm". Even if side one of your card was ok, you need to run it through the data write cycle here. This will instruct the card to add the "crd" instruction. If you don't run side 1 first, it will put the "crd" instruction on side 2. Finish by running side 2 through the card reader. If the card gets jammed, you may have added too much tape to the corner. You will need to use another card to help push the card through.

That's it. Switch the program switch back to "run" and try the card. If it works, remove the tape. It has helped me save several pre-recorded cards. Please, I do not want to accept any liability if you damage your card reader or calculator. Please perform this operation at your own risk.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

How to un-private an HP-41 program

Posted by [Steve \(Australia\)](#) on 22 June 2000, 6:05 a.m.

First, some background. (If you don't need this, just skip down a bit to where the method is described)

The HP-41 card reader (and other mass storage devices) allow you to save a program as "Private". This has the effect of preventing someone making changes to the program.

It is the END instruction at the end of a program that holds the flag that a program is private. It is this that we have to remove to un-private the program.

When it was devised by HP, they did not provide a way to un-private a program. So if you accidentally saved a program as private, or corrupted the cards with the normally saved one then you were left in a bit of a dilemma.

However, BUG 9 comes to the rescue. For more information, read P621 of "Extend Your HP-41" (The not so little red book).

BUG 9 provides the method used to bootstrap the byte-grabber into the key assignment registers, thus making Synthetic programming a rather simple process.

This method is very similar to that procedure. I came across this while playing around one day. I make no claim to it being my discovery, there were far more gifted, industrious, and just plain geeky people ahead of me that (no doubt) found it before me.

HOW TO DO IT!!!

This assumes that the private program is the last in memory. This procedure worked on my HP 41, but NO WARRANTIES ARE EXPRESSED OR IMPLIED. If your card reader turns into a potato, it's your own fault!

GTO.. (this packs memory, but more importantly, puts you at the end of it)

PRGM (to put calc into Program mode)

LBL "XXX" (Enter this)

GTO.. (This places an END at the end of the program you just keyed in.)

CAT 1 (Stop when LBL XXX shows in display)

SST (Displays LBL "XXX")

SST (Displays .END. REG XX)

ALPHA (Place calculator in Alpha mode)

CLX (display is 4094 END)

BST (display is 4093 LBL"XXX")

BST (display is 4092 END -- this is the end that would normally trigger the display of PRIVATE, but here it doesn't)

CLX (display is 4091 xxxxxx -- last line of private program. We just got rid of the pesky private end)

SST (display is 4092 LBL"XXX")

CLX (removes label)

ALPHA (out of alpha mode)

GTO .. (puts a normal end on the program, and resets the numbers back to what you would expect)

And that's it.

However HP calculators must have inbuilt defence mechanisms. My calculator crashed before I started to try this, and afterward my internet connection failed, then my computer crashed as I reconnected. This is the second (and hopefully last) time I write this procedure :-)

And now that's three times. The HP defence mechanism is really strong!

And now it's 4 times as I type it in here.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Recovering shorted battery packs

Posted by [Steve \(Australia\)](#) on 19 June 2000, 2:40 a.m.

WARNING - this technique is not for the faint-hearted, and involves some personal risk. But as long as you're not a few snags short of a barbie (i.e. you have you're wits about you) it can be done safely.

What you require:

- 1) a dead NiCad battery pack. (Say the one out of the HP-97 calculator). It's important that the nicads be in good condition (not leaking).
- 2) a high voltage low impedance power supply. Something that can deliver say 200 volts at 20 amps should be fine. Seriously!
- 3) a multimeter (preferably digital) capable of reading DC voltages between tens of mV and 350 volts.
- 4) something that will charge the pack.
- 5) No fear of zapping sounds.

Nicads that have been left for a while tend to internally short. The typical symptoms are a pack that charges, but only gets to a voltage of (say) 2.75 volts instead of (nominally) 4.8 volts.

If these cells are subjected to a high current, but short duration pulse, the internal short can be removed.

I use a disposable camera flash unit. I solder a pair of heavy duty well insulated wires across the capacitor. The other ends are tinned. Note that figure 8 cable of some sort (preferably mains rated) is best.

PRECAUTIONS: Never place yourself in the position where the capacitor can discharge across your chest. In other words, don't hold the leads one in each hand. Similarly, don't measure the voltage by holding one probe in each hand. If you're paranoid, keep one hand in your pocket. The voltage is only 350 at a max, but the cap can deliver in excess of 50 amps!!!! While you'd have to be very unlucky to be killed by it (the total energy is quite low) it would scare the crap out of you. Also be careful that the battery doesn't rupture (unlikely, but remotely possible).

Not scared?

First take the flash unit out of the camera. Be careful, the charge on the cap is enough to kill you if correctly applied! Take out the battery and short the capacitor (the big -- usually black -- cylindrical object with 2 wires coming out of one end) with an insulated screwdriver. BANG! (if charged, it really does).

Some of the capacitors have markings on them. A 330 (or 350) volt 160uF capacitor is common and the right value. These are about 1/2 inch in diameter, and about 1 1/2 inches long. There are some smaller ones in newer cameras that are about half the size, but these should be OK too.

Now solder together whatever contacts are required to make the thing charge. As soon as you insert a battery it will start to build up a voltage. Then solder the wires onto the capacitor. At the same time you can remove the wires that go to the trigger -- or just let them flap around.

At this point, if you insert a battery, you have a cattle prod. I would NOT suggest trying it out in this mode!

Insert the battery for about 2 seconds. You should hear a whine, and hopefully not zap yourself. Take the multimeter and measure the voltage across the wires. It should be between about 30 and 100 volts, but this depends on the state of the battery, and the type of flash unit.

I have found that a charge of around 100 to 150 volts works well (and it's not critical). (Use a lower voltage for smaller batteries)

Charge up to this voltage, and place the +ve wire on the +ve terminal of the battery, and the -ve lead on the -ve terminal. BANG! This will leave a small mark like a spot weld on the terminal -- don't worry.

Now place the battery on charge for 5 minutes, and check the voltage. It's probably OK. If not, open the battery pack and check each cell. A cell with less than 150mV after a short charge is certainly dead. Give it a blast!

It's also a good idea to completely discharge the pack, cell by cell, to ensure that all cells will be charged equally. Another good check is to charge, then use the pack, then check that all cells have the same (or very nearly the same) voltage across them.

This technique is faster and cheaper than rebuilding a pack, but PLEASE be careful.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Repairing the hp 82104A Card Reader

Posted by [Dave the "lostrabbit"](#) on 25 May 2000, 12:05 p.m.

Pictures by [Ty Rogers](#)

Many of us these days with the 82104A card readers for the hp 41 series of calculators have been having problems with the driven gear roller becoming "gooey" and therefore rendering our beloved readers inoperable. I have repaired quite a few of these, and now I have decided to publish my procedures here so that you, the do-it-yourself type, can repair yours.

The disclaimer!

I am not affiliated with hp, and any attempts you should make to repair your equipment is based solely on your judgement. That being said, lets jump in!

The tools

Ok, first, find yourself a a good place to work, a drafting table is a great place. Get the surface clean, after all, we will be performing surgery here with MANY small parts. A first time repair may take a few hours, so keep that in mind and send the kids to the movies etc.

The tools you will need include:

A very small philips head screwdriver

A not quite as small philips head screwdriver

A curved pair of hemostats

An x-acto or utility knife

Several cotton swabs

A bottle of isopropyl alcohol (NOT beer!)

Material for the repair: either a small piece of model airplane fuel line (the pink tubing with the small inside diameter OR a pair of O rings, somebody else has stated that they are size 006, dimensions are 2.9mm OD, 1.78 ID.UPDATE! I have yet to find a "size 006" O ring, the ones that I have used are english measure 1/4" OD, 1/8" ID, which works out to be something like 6.35mm OD and 3.18mm ID.

Ready?

Ok, first we need to remove the label on the bottom of the card reader. Holding it in your hand, very gently lift it up at the edge with the utility knife. A little bit at a time until you can get it to release. Careful, the knife is sharp and we are not after an oyster here.

Now that the label is off, you now see three screws. Direct your attention at this point to the front of the reader, where

it fits into the calculator. Here you will find 4 very small screws. Remove the 2 lower screws, one on each side, and put them in a safe place.....an empty plastic pin box works well.

Now back to the 3 screws under the label. The one you want to remove is the one at the rear of the reader. The other 2 hold the drive slot assembly to the bottom half of the unit, and you aren't ready to remove those yet. Having gone this far, you should be able to lift the top half of the unit off the bottom half. Do this over the table as things may fall out! Work it, a little at a time and you will get it. Note the way the latches are assembled, make a sketch if you need to.

[Picture of the opened card reader](#)

Topless!

Ok, you have the top off and should now have the top half, the 2 latches each with a piece of metal strip that act as springs, and the front "curved" piece of plastic that fits against the 41 of your choice. Put all this aside.

Next, we want to remove the drive slot assembly from the lower half of the unit. First, I like to remove the 3 screws that hold down the circuit board to the slot assembly. Be VERY careful here as once you remove these, the circuit board will come off the vertical pins that make the contact and you will find a pair of long copper strips shaped like a fork below it. Do not bend these! Remove the last pair of screws that were under the label. Now the assembly is loose.

[Picture showing drive slot assembly](#)

[Picture showing drive assembly removed](#)

Close quarters!

We are getting there! Ok, I like to remove all the wired connections at this point so I can get to things. Its up to you whether or not to do this, but if you are careful and make a sketch of things it is the easiest way to go in the long run. Note the position of the wires on the left side. There should be 5 of them, with the order from TOP to BOTTOM as such: RED
YELLOW
BLUE
ORANGE
BLACK

As for the power supply to the motor, there are 2 wires on the far left, RED and BLACK; NOTE that the RED wire is closest to the front of the assembly, i.e. the calculator end of the unit.

Here's where the hemostats come handy, as the wires are a "friction" fit, they slip into the connections. Carefully pull each wire out with the hemostats and the drive slot assembly should be free.

Patience is virtue

Ok, directing your attention to the drive slot assembly, notice that it is held together by 2 screws, one on the top right, and one on the bottom left, very close to the worm gear etc. That's why it is easier in my opinion to unwire the thing!

Remove these 2 screws OVER THE TABLE and then separate the top half of the drive slot from the bottom BEING VERY CAREFUL!! A few things that may happen here is that a small curved piece of copper may take flight, an EXTREMELY SMALL white plastic roller may leap for the carpet, or another small copper strip may jump out. This last one do not bend or else you will be sorry! I believe it is a type of spring to hold the card down as it passes over the tape head.

Yee gad, the GOO!!

If your card reader was putting the "goo" on your cards, you now can see the problem! More than likely your inner drive slot has lots of goo all over the place, with a good build up around the magnetic head. You say "gee Dave, you are correct!" (99% of the time, anyways!)

Picture showing read/write head and original wheel

Ok, put all those little pieces; the TINY white roller and the curved copper strip in your pinbox or whatever you are putting the parts in. You do have something to put them in, right?

To remove the driven gear with the gooey roller, use the hemostats. You will find a plastic pin with a slot in the end of it that looks like you need a screwdriver to remove it. But you don't! It is a friction fit, and a little twisting action as you pull with the hemostats will remove it. The pin is out, the driven gear with the gooey roller will now come out of the assembly through the top. Put the pin and the gear in your storage place be it the pinbox or whatever for now.

Now to clean. Use cotton swabs and the alcohol to clean the goo. Get it all. Use quite a few and make sure ANYTHING and EVERYTHING with goo has been cleaned. Be VERY CAREFUL around the tape head, in fact I like to clean that part last with fresh swabs and alcohol, kind of like cleaning a cassette deck back in the 80's. HA! Clean the little white roller as it will be gooey also.

All clean?

Now that everything is clean and there is ABSOLUTELY no trace of the offending "goo", we are ready to go back together.

Lets direct our attention to the driven gear, the one with the gooey roller. You need to clean off the goo, and there you will find an aluminum shoulder. Having made the decision whether to use the fuel line or the O rings it is now time to fix the problem gear. If you use the fuel line, slip it on then cut it off, chamfer the outer edge ever so slightly as to keep it from binding. It is not hard to do, just be patient with it. The O rings I do admit after jousting with Steve, offer an easy way to go. I dismantled one of the ones I had previously repaired with the fuel line and tried the O ring method. It works fine, although I could not detect (from the sound of the reader in operation) any benefit on the mechanicals.

Picture showing read/write head with o-rings

If your gear is not totally gooey, the stuff will be tough to come off. Use the knife and get it all off. Apply the material of your choice and we are ready to go back together. The gear is clean, right?

A few things I should point out beforehand. Check the condition of the worm gear on the motor. If you have goo there, clean it with a swab and the alcohol. Get it as clean as you can. Also, I like to check the fit of the worm gear on the shaft by touching the power supply wires to a single N cell battery and then while the motor is running apply a touch of the finger to insure it stalls. If it is slipping apply a touch of cyanoacrylate glue (crazy glue) to the shaft making sure it doesn't run down into the motor! Also, apply lots of alcohol via the swab to the end of the worm gear where it fits into the plastic housing, i.e. the bearing area. There is more than likely goo here also.

Back together!

Take a break, go wash your hands. Cleanliness is everything here! Ok, insert your repaired driven gear into its place, line up the hole for the plastic pin and then put the pin in. It should slide in with out much hassle. Check to see that it is not binding with the worm gear, there should be a little rocking action between the gears.

Put the small white roller from whence it came back in its place in the top half of the slot assembly. Place the curved copper strip in its place in the bottom half. Now, very carefully attempt to put the top half of the drive slot onto the

bottom half. A good approach is to hold both on edge and then fit them together using the locating dowel pins at each end. Beware the fact that things may fall out, especially the tiny roller, and that the copper strips must fit in the right places! It may take a few attempts to get this part right, but be patient! It will come together. Holding the two parts together replace the 2 screws that you removed a long time ago. One at top right, one at lower left on the underside.

Be careful with the spring contacts, the fork shaped ones. Again, do not bend these!

Having that part re assembled, put the circuit board that makes the contact with the fork shaped springs back on top. I like to start all of the 3 screws then tighten down the center one first, then the outer 2. Not real tight!

Getting there!

Ok, now its time to rewire the jewel. Use the diagram I gave you earlier, and with the hemostats very carefully fit the 2 wires to the motor into their connections.

Next, place the drive slot assembly in its position, making sure the upper circuit board fits down and over the gold vertical contacts. Line up the holes, and fit it down over the contacts. Then fit the bunch of wires on the left side to their respective connections. Using the hemostats, they should just slip in nicely. the wires are tinned with solder, and should not become frayed if you are careful.

Replace the 2 screws that were under the label that hold the drive slot assembly into place.

Testing, 1,2,3

Ok, time to see what goes! Fit the card reader to your calculator and give it a try. Insert the card and see if you have done things right! It should work, but if it doesn't there are a few things that could have gone wrong. If the card starts to be pulled through but hits a "brick wall" then chances are you did not get the copper strip at the tape head end in its place, and it is hanging up there. No problem, just dis assemble and re assemble. Another problem, if it does not come to life when you insert the card then chances are that your copper strips shaped like a fork got bent or misaligned (I told you to be careful, didn't I?) You can check this by watching the deflection of these as you insert the card. They are operated by the 2 small plastic cams at their ends. They should lift up and make the contact with the upper circuit board.

IT WORKS!!!! YIPPEE!!!!

Prepare to be frustrated!

Ok, now it is time to button things back up. This part can be tough, but after several different methods, I think this one has proved to be the best.

Put the card reader assembly aside, and reach for the top piece, the latches and their respective springs(the strips of metal). Note that one of these strips has a plastic or tape kind of coating, usually yellow but sometimes white. It goes on the LEFT so as to not short the components.

Now, holding the top in your hand, put the latches with the spring plates into position. The springs fit behind the vertical metal strip that the very small screws go through.

Now, holding the top with the latches and spring plates together in one hand, put the card reader assembly in your other hand and fit the top down over the bottom. Align the latches so their pins fit into the holes of the top and bottom halves. Get it together, but not quite all the way. The spring plates should be swinging in the breeze right now, that's how they should be.

Last step, get the curved piece of plastic and noting that it has a couple of tabs on the inside that the spring plates fit

between, very carefully fit it between the latches and nearing position, fit the bottom of it into place making sure the spring plates fit between these tabs, then push forward with the plastic piece and fit it in at the top. this usually takes a few attempts, but I really do think it is the way to fly!

Replace the long screw at the bottom where the label was, and the 2 very small screws at the calculator end, apply the label and you are done!

Oh, one more thing!

I hope you now have a WORKING card reader, and that I have helped you out! All I ask is that if you did use this material and it solved your problems (with the card reader that is) that you promptly send me a picturesque postcard from your part of the planet. Long live the 41!

Dave, aka "the rabbit"

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)



HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-41/PC User-Code Transfers

Posted by [Leo Duran](#) on 24 May 2000, 12:16 p.m.

This article has been moved to the [Software Library](#)

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-41C "Decompiler" and "BarCode Maker"

Posted by [Dan McDonald](#) on 14 May 2000, 1:11 p.m.

This is an Excel file with macros that will produce-on a PC running Excel-a listing and a barcode of an HP-41 binary program file. The HP-41 Binary files are available online at the swap-disk archive (<ftp://ftp.math.jyu.fi/pub/hpil/swap/>) or on a LIF disk somewhere. My [previous article](#) on the HP Museum site shows how to get these files to your PC and how to remove the extraneous LIF header data from them.

Getting a fast listing from the PC is essential to figuring out what some of these files are. Getting a fast BarCode of these files is just kind of fun and something I wanted to see if I could do.

The files I've included, and which are needed for this project are:

- [41_ListA.xls](#) (Excel file with macros)
- [41bar3.ttf](#) (font file, includes "41C BarCode" font for PCs. This must be installed on the PC in the appropriate place and manner.) **OR**
- [41bar.sit](#) (Stuffit file, includes "41C BarCode" font for Macs. This must be installed on the Mac in the appropriate place and manner.) Rob Frohne made this Mac font from Dan's PC version.
- [nim41](#) (an example of a 41C Binary file. LIF info: Type = E0 80, Implementation = 02 9A 00 20)

To download the files above in Netscape, right click on each link and click on "Save Link As..." Alternately, you may find it easier to download all of the above files in this [zipfile](#).

Notes:

Special (non-DOS) characters are not implemented yet. I've used the words , , and to indicate them. If the user doesn't like these, they are welcome to change them in the appropriate Excel cell.

XROM functions are not implemented yet. They appear as "XROM xx,yy" and it is up to the user to figure out which ROM these are from and what the command is. The correct "xx,yy" are supplied for every XROM function, of course, it's just that it's hard to tell from that which ROM it's coming from (xx) and which command within that ROM it is (yy).

I'm slowly working on some other fonts, including a 41 Display font and a 82143/82162 dot-matrix printer font.

Error trapping is almost non-existent. Be wary of source files or paths spelled wrong and duplicate Excel worksheet names, and more. Some of these will put Excel into an infinite loop, others will just stop the macro from running.

The operator interface still needs some work, but it's not too bad.

Instructions:

1. Have the 41C Binary file of interest available on the machine somewhere.
2. Open the Excel file [41_ListA.xls](#) Go to the worksheet "41_List" you should see 2 colored boxes ("Make

Listing!" and "Make BarCode!") near the top rows.

3. Type the name of the file of interest (include the path) somewhere, and make it the current Excel cell. You actually don't have to do this, but it's nice to have it somewhere. If you don't do this, you'll have to type it all into a dialog box, and then do it again if something goes wrong.
4. To get a listing:
Click on the "Make Listing!" colored box (it's actually a "button") This puts a text file of the program listing into a location you specify
5. To get a BarCode:
Click on the "Make BarCode" colored box. This puts a printable barcode program listing into a new Excel worksheet (you specify the name)
It also puts the information into a non-fonted text file which you can then later load into another program and give it the correct font...this is not as nice as the new Excel worksheet, and is pretty much unnecessary.

Misc:

These programs were written for fun not profit, so all are welcome to use, modify, or distribute them, but please don't sell or charge anybody for using them!

If there are any questions about the usage or implementation of these programs, please let me know.

Happy calculating.

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

AWK Translator for Keycodes

Posted by [bill duncan](#), [ve3ied](#) on 23 Apr 2000, 1:08 a.m.

Reaching Out

I have a number of calculators I can reach for.

I have three generations of the venerable HP-41C (a C, CV and a CX), HP-16C, HP-951x, and an HP-48GX. (I also used to have an HP-15C, but someone else reached for it when I wasn't looking...)

So which one do I reach for the most? It's the old 80/20 rule. I'm not a "rocket scientist", so I don't spend my days doing much calculus. My trusty old HP-29C does what I need most of the time. It's the oldest design, and the least capable machine I have. I guess I just like the solid feel of the Woodstock series. The HP-41's seem frail by comparison and the HP-48GX is just, well, *big*.

The Inspiration

While pulling out a pad of graph paper to copy down (yet again) the routines which happened to be in the HP-29C (sort of like a manual backup), I decided that this was silly. I'm surrounded by Linux systems. Why copy the contents of the calculator to paper?

Of course, like all the old "keystroke programmables", the keycodes that needed to be translated back to English. *Boring*. Why not get the computer to do that too?

The Program

I've designed the following program such that theoretically it can also be used for calculators other than the HP-29C. If I ever get a replacement for my HP-15C, I may even test that theory... (*Anyone have a line on one for a good price?*)

It was written in AWK which is readily available if you have a Unix or Linux based system. (Some Unix systems will call this "nawk", or you may have "gawk" or "mawk" installed.)

If you are still using a Microsoft or Apple based system, there are versions of AWK available. I recommend "[mawk](#)" or "gawk". (*And I really recommend switching to [Linux](#)...*)

There is nothing spectacular about the program. It's fairly simple, and uses a separate symbol table which might be changed for another calculator. I thought that I might improve the program by doing page headers etc., but that's left as an exercise for another day (or for someone else). [Email me](#) if you have suggestions or improvements you'd like to make.

The awk script is available [here](#) and the symbol table for the HP-29C is available [here](#).

The program will convert simple keycode listings (without the line numbers) such as the following snippet:

```
# -----
# %CH - Percent change
# -----
15 13 01
21
41
14 73
71
33
02
61
15 12
13 01
...
...
```

Here is a sample output formatted listing of what I happen to have in my calculator now. The lines starting with an octothorpe (hash mark, "#") are just comments which I placed in the input file and come out the other end unaltered. The program generates the mnemonics and the line numbers, so that if I need to rekey the programs it is relatively easy. Certainly much more fun than looking at keycodes.

```
# @(#) HP-29C contents as of April 21, 2000
# @(#) $Id: 29c.html,v 1.1 2000/04/23 06:36:42 bduncan Exp bduncan $
#
# Author: Bill Duncan (except where noted)
```

```
# -----
# %CH - Percent change
# -----
01 15 13 01 LBL 1
02 21 x<>y
03 41 -
04 14 73 LASTx
05 71 /
06 33 EEX
07 02 2
08 61 *
09 15 12 RTN
10 13 01 GTO 1
```

```
# -----
# %T - Percent of total
# -----
11 15 13 02 LBL 2
12 15 74 1/X
13 15 21 %
14 15 74 1/X
15 15 12 RTN
16 13 02 GTO 2
```

```
# -----
# FIB
# -----
# number in both x and y, flashes successive numbers in the series.
# Standard Fibonacci sequence starts x=1, y=1.
# Try y=1, x=3 which is an interesting sequence in that the numbers
# approach integer powers of phi.
#
17 15 13 08 LBL 8
18 51 +
19 14 73 LASTx
20 21 x<>y
21 14 74 PAUSE
22 13 08 GTO 8
```

```
# -----
# parallel resistors
```

```
# -----
23 15 13 04 LBL 4
24      15 74 1/X
25      21 x<>y
26      15 74 1/X
27      51 +
28      15 74 1/X
29      15 12 RTN
30      13 04 GT0 4
```

```
# -----
# degrees C to F
# -----
#
31 15 13 05 LBL 5
32      09 9
33      61 *
34      05 5
35      71 /
36      03 3
37      02 2
38      51 +
39      15 12 RTN
40      13 05 GT0 5
```

```
# -----
# fib #2
# -----
# related to fib sequence
# phi [(SQR(5)+1) / 2] stored in REG 05 before running!
# key number in X, good approximation out to 35 or so...
# this sequence comes close to phi^n
# simple change to get standard Fibonacci sequence
# just change the last instruction from a "+" to "-" and divide by sqrt(5)
#
41 15 13 06 LBL 6
42      24 05 RCL 5
43      21 x<>y
44      14 64 Y^X
45      14 73 LASTx
46      24 05 RCL 5
47      01 1
48      41 -
49      32 CHS
50      21 x<>y
51      14 64 Y^X
52      51 +
53      15 12 RTN
```

```
# -----
# quadradics
# -----
# from 29c applications book page 6.
#
54 15 13 07 LBL 7
55      23 00 STO 0
56      22 RDN
57      21 x<>y
58 23 71 00 STO / 0
59      71 /
60      02 2
61      71 /
62      32 CHS
63      23 01 STO 1
64      31 ENTER
65      15 63 x^2
66      24 00 RCL 0
67      41 -
68      74 R/S
69      15 41 x<0?
70      13 00 GT0 0
71      14 63 SQRT
72 23 41 01 STO - 1
73      21 x<>y
74      51 +
75      24 01 RCL 1
76      14 73 LASTx
77      15 51 x>0?
```

78		22		RDN
79		22		RDN
80	23	71	00	STO / 0
81		24	00	RCL 0
82		13	03	GTO 3
83	15	13	00	LBL 0
84		15	64	ABS
85		14	63	SQRT
86			21	x<>y
87	15	13	03	LBL 3
88			74	R/S
89			21	x<>y
90		13	03	GTO 3

```
# -----
# H.MS+
# -----
# numbers in x and y both in H.MS format
# LASTx is lost. Ah well...
#
91 15 13 09 LBL 9
92 15 72 ->H
93 21 x<>y
94 15 72 ->H
95 51 +
96 14 72 ->H.MS
97 15 12 RTN
98 13 09 GTO 9
```

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

The continued fractions algorithm (was: Re: Converting a decimal # to a fraction #)

Posted by [rupert9000](#) on 22 Apr 2000, 6:09 p.m.

Well, for my current practical purposes I developed the simple algorithm previously posted, but now I have seized the opportunity to re-implement a more efficient algorithm based on the continued fractions. It is well explained, for practical purposes, on the HP67 Mat I Pac manual (afterall this is the HP Museum...):

for any given number, let's say decnum, the continued fractions algorithm can generate a converging serie of fractional approximations of it. Each fractional approximation is computed as a numerator n(i) and a denominator d(i) as follows:

$$n(i) = a(i) * n(i-1) + n(i-2) \quad d(i) = a(i) * d(i-1) + d(i-2)$$

The following values are assigned by definition: $n(-1) = 0$, $d(-1) = 1$, $n(0) = 1$, $d(0) = 0$.

The values for a(i) coefficients are computed as follows:

$$a(i+1) = \text{int}(x(i) / y(i)), \text{ where } x(i+1) = y(i) \text{ and } y(i+1) = x(i) - a(i+1) * y(i).$$

Again, the following values are assigned by definition: $a(1) = \text{int}(\text{decnum})$, $x(1) = 1$, $y(1) = \text{frac}(\text{decnum})$.

For a reference on the web look at <http://archives.math.utk.edu/articles/atuyl/confrac/>.

Here is a listing (on this site it might not be properly displayed), written in plain BASIC because it allows to easily test an algorithm, is self-explaining and easily adaptable to any kind of calculator/computer.

```
REM Continued fractions algorithm
```

```
DIM a1, a2, x1, y1, x2, y2 AS DOUBLE
```

```
DIM dm1, d0, d1, nm1, n0, n1 AS DOUBLE
```

```
DIM decnum, delta, f AS DOUBLE
```

```
DIM i AS LONG
```

```
CLS
```

```
PRINT ""
```

```
INPUT "Number "; decnum
```

```
nm1 = 0
```

```
dm1 = 1
```

```
n0 = 1
```

```
d0 = 0
```

```
a1 = int(decnum)
```

```
x1 = 1
```

```
y1 = decnum - a1 : REM y1 = frac(decnum)
```

```
i = 1
```

```
DO
```

```
a2 = INT(x1 / y1)
```

```
x2 = y1
```

```
y2 = x1 - a2 * y1
```

```
n1 = a1 * n0 + nm1
```

```
d1 = a1 * d0 + dm1
```

```
f = n1 / d1
```

```
delta = ABS(decnum - f)
```

```
PRINT "i="; i; " N(i)="; n1; " D(i)="; d1; " N(i)/D(i)="; f; " Delta="; delta
```

```
PRINT " press any key..."
```

```
DO
```

```
LOOP WHILE INKEY$ = ""
```

```
a1 = a2
```

```
x1 = x2
```

```
y1 = y2
```

```
nm1 = n0
```

```
dm1 = d0
```

```
n0 = n1
```

```
d0 = d1
```

```
i = i + 1
```




HP Articles Forum

[[Return to the Index](#)]
[[Previous](#) | [Next](#)]

HP-IL Files: Where, What, Why, How, etc.

Posted by [Dan McDonald](#) on 7 Apr 2000, 12:34 p.m.

This is a piece that I have been working on for some time, and will probably never "finish," so I've decided to put it out to the public. If you have any questions about this, or would like any additional programs or software mentioned in the text, please email me at danm@hevanet.com

Thanks, and happy reading and HP-IL'ing.

Dan McDonald

danm@hevanet.com

April, 2000

Rev. 2.1

Ancient Data – Modern Machines: Handling HP-IL Files on Today’s PC

Archive, store, transfer, and copy HP-IL files to and from your modern DOS computer, without the HP-IL/PC Interface Card (HP 82973A). Reach into the past; access and use the wealth of data and expertise that now quietly resides in the "Swap Disk" internet ftp archive. Thousands of files and programs exist for the HP-41, 71, and 75 and are easily available for (almost) immediate use.

Hardware required:

HP-IL capability for your HP-41, 71, or 75.

HP-IL disk drive (HP 9114B)

Any DOS-compatible PC (including Pentium)

Software required:

Public domain programs from the internet

Optional small programs (available on-line as well)

Table of Contents:

Introduction: *

Scope: *

Background: *

Today’s Status: *

How is this useful? *

Technical Background Information: *

Logical Interchange Format (LIF) Storage Structure and Information: *

The Files. *

The LIF Drive Directory. *

The Individual LIF Directory Entry. *

The HP 82973 PC/HP-IL interface Board: *

HP 82973 file storage: *

HP LIF2DOS Software *

Something Useful: Get and Run a file from the Swap Disk Archive *

Something Else Useful: Distribute Your Own LIF Files to Friends and Enthusiasts *

So That's That *

Important Information: *

Useful Information, or "What do Those Files DO, Anyway?" *

A word or two on Formatting LIF Disks: *

Disks formatted with the HP-41: *

Formatting disks with the PC and LIFUTIL: *

A word or two on acquiring hardware: *

References and Resources: *

PPC – LIF Storage Info (partial listing) *

CHHU – LIF Storage Info *

Swap Disks *

The Internet *

Acknowledgements and Special Thanks to: *

Appendix I – Basic LIFUTIL Instructions *

Appendix II – Automatic Extraction of Important File Info *

Appendix III: LIF File Types *

Appendix IV: HP-IL Ebay Prices, Summer 1999 *

Appendix V: HP-IL Device Info *

Appendix VI: Summary of On-Line Files and Information and Programs Referenced: *

Introduction:

The purpose of this paper is to describe how to transfer HP calculator files between DOS and HP-IL, without using the specialized PC/HP-IL interface board (HP 82973A). Background information will be presented to explain the context of this topic and show its relevance. Applications of these techniques will also be shown to illustrate their usefulness.

Scope:

This paper refers specifically to files generated or used by the HP-41, HP-71, and HP-75 series of calculators. The same general procedures apply to other LIF files, generated by other HP equipment, but any subtle nuances of these files are not explored or explained.

Background:

The HP-41 calculator was introduced in 1979, long before desktop personal computers became ubiquitous. As an alpha-numeric, programmable,

expandable calculator, it was a very powerful computational tool and became immensely popular. Some of its features that are important to this topic include:

Could do "long" programs

Had Input/Output capabilities

Expandable, to accommodate future, as-yet-undeveloped hardware

A devoted user community developed around this calculator, in many ways an extension of the devoted user community that had developed around earlier HP calculators. Specialized applications and programs were created to exploit the calculator's capabilities. The early means of distribution of these user-developed programs included the following:

Source-code printouts

Magnetic Cards

Bar Codes

Customized ROM

Each of these distribution methods has its advantages and disadvantages.

Advantages	Disadvantages
Source Code Listings:	
Cheap Easy No additional hardware needed except perhaps a printer for one person to create the listing	Each user would have to re-key in the entire program. Lengthy and prone to mistakes.
Magnetic Card Storage	
No need to re-key in the program – quicker and mistake-free	Requires additional hardware (Card Reader/Writer) to create or use the cards. The Card Reader/Writer became notorious for consuming batteries Could require several magnetic cards per program
Bar Code Creation	
Cheap and easy to distribute (photocopies) No need to re-key in the program – quicker and mistake-free	Requires additional hardware (Bar Code Reader) to read the code. Time-consuming to create the bar code. Requires additional hardware (ROMs, printer or plotter) to create the bar code. Plotting bar code tended to quickly wear out the tips of expensive plotter pens.
Customized ROM	
Could include a lot of information with one ROM module No additional hardware required (except the ROM itself)	Expensive to develop and produce Since it is Read-Only, the contents cannot be easily changed

Table 1: Advantages/Disadvantages of Various Source Code Distribution Methods

As time went on, new hardware based on the Hewlett Packard Interface Loop (HP-IL) was introduced that provided additional options for storage and transfer of programs. This included a cassette tape drive at first, followed later by a floppy disk drive, as well as other devices.

The advantages and disadvantages of the tape and disk drives are summarized in this table:

Advantages	Disadvantages
Digital Tape Drive	
Could store much information on one tape Relatively easy to distribute No need to re-key in each program.	Additional hardware required (HP-IL Interface adapter, and tape drive itself). Slowish access time. Much winding and rewinding of the tape in order to read individual programs
Floppy Disk Drive	
Could store much information on one disk Relatively easy to distribute No need to re-key in each program. Much faster than the tape drive	Additional hardware required (HP-IL Interface adapter, and disk drive itself).

Table 2: Advantages/Disadvantages of Digital Tape Drive and Floppy Drive

The user communities would hold regular meetings and conferences where ideas and progress could be discussed, and programs exchanged. Even though additional hardware was expensive, there would usually be somebody amongst those gathered who could bring hardware to share, and programs could be disseminated readily at these meetings.

Additional calculators/computers were introduced that could use the HP-IL hardware: the HP-75C/D, and the HP-71B. These machines had strong followings as well, and masses of specialized programs were created to support these platforms. The files created for these machines became larger and more complex. They included large BASIC programs, and special Logical Extension Files (LEX) that greatly expanded the capabilities of the HP-71B. The LEX files took additional hardware (ROMs) to create, but once compiled could be distributed and used without any extra hardware necessary.

During these times, a user "Swap Tape" program was developed. New programs would be collected and copied onto tape, then the tape could be copied and distributed to users. At user conferences, each person would supply their own tape (and contribute programs), all the programs would be copied onto a master tape which would then be re-copied back onto the users' supplied tape to be taken home.

The Swap Tape program evolved into a "Swap Disk" program, where the same thing would happen using disks for media instead of tape. A single disk would contain programs for the HP-41, HP-75, and HP-71.

HP produced test equipment that used the same disk formats as the HP-IL drive. This test equipment sold very ell, but HP-IL disk/mass storage devices never becam big, cheap, or widespread.

Later on, HP developed a card for the PC that would provide access from the PC to HP-IL devices. This card is called the HP 82973A. Having this card should allow the owner to easily transfer HP-IL files to DOS and back, making the procedures in this paper unnecessary. This card allows the user to do other interesting things, including using the PC keyboard as a keyboard for the HP-71B. (It should be pointed out that HP-71B BASIC files can be composed as text files on the PC, then transferred to the HP-71B using the techniques from this paper, and compiled into BASIC there.) While these 82973A cards are occasionally available on ebay, they were never widely popular, remain somewhat rare, are still relatively expensive, and come with a big warning. The warning is: *these cards were designed for old-era PCs (16 MHz speed), and are reported to be unstable at best when installed in a modern PC.*

Eventually with the progress in computing and handheld device technology, and for various other reasons, the user community for these devices has changed. Published journals and organized clubs no longer exist. Even the venerable HP Journal has ceased publication and moved "on-line." The years of dedicated enthusiasm by a large user community for the older machines have left behind a legacy of published documentation and archived specialized programs.

Publications and journals of the era from Hewlett-Packard and the user community have been preserved and are available from Jake Schwartz, at <http://www.jakeschwartz.com/>. His set of CD-ROMs contains over 2,000 pages of tidbits, programs, history, insights, drama, and intrigue. Believe it or not, this can be fascinating if not essential reading for those interested in calculators, calculator history, or just trying to use and understand their good old calculator(s).

The "Swap Disk" program, although "dead," has been archived and lives on-line at <ftp://ftp.math.jyu.fi/pub/hpil/> and subdirectories. Thousands of files for the HP-41, 71, and 75 are stored there to be downloaded and used.

Today's Status:

The HP-75C/D, apparently never very popular, is even less so now. The HP-71B has some life today as an excellent handheld implementation of BASIC and a unique and powerful calculator. The HP-41 series of calculators remains popular in the engineering and survey worlds as an excellent easy-to-use programmable RPN calculator. Continuing the tradition of independent user groups, there is strong widespread user support and dedication to HP's modern calculators (HP-48 series). Hewlett Packard released a new calculator (HP-49G: in stores mid-August, 1999) which initially received mixed reviews from the user community.

Today's computing world is filled with cheap and plentiful computers with capacious hard disk storage. Email and networks are pervasive and available. Calculators are no longer the only option for the center of the personal computing universe. The old calculators, however, still have just as much computing power as they ever did, and what's more, most of the extra hardware for them that was way too expensive at the time is still available now but is much more affordable.

So for a moderate investment, the hobbyist of today has the option of owning top-of-the-line equipment from an earlier era, equipment that may satisfy or exceed all of their needs. A person may have, for example, an HP-41 calculator that they want to "trick" out, or use to perform some type of special function. This person may be interested to know that perhaps what they want to do has already been done and resides peacefully in an archived file easily accessible by ftp. This paper will help them find out if that is true, and show them how to get and use that file.

How is this useful?

There are many people who still use and enjoy their old calculators. If this seems unbelievable, a check of ebay prices for HP calculators and HP-IL equipment will show that there is some value left in these devices. Perhaps also of interest are the ebay prices of TI calculators. It should be found that the TI calculators command a much smaller price than HP calculators do. Reasons for this are open to debate, but the point that people collect (and presumably, some are collected to be used) and pay good money for HP calculators should be well taken.

There are several advantages to the HP calculator user of being able to transfer files between the HP-IL file system and the PC/DOS environment. Some of them are listed below:

1. Mass Storage of HP-IL files. Important files should be stored outside the calculator somewhere, as calculator memory can be lost or inadvertently cleared. The HP-IL storage devices are, small, old and slow. DOS storage devices are ubiquitous, faster, and big. If HP-IL files could be easily kept on DOS instead of old and slow HP-IL, they could be stored, backed up, and handled easily in bulk on a reliable hard drive
2. Distribution of files. DOS files can be emailed around or stored on-line for easy retrieval. This avoids the old problems of messing with cards or bar code, or sending disks or tapes via surface mail.
3. Access the Swap Disk ftp archive. Thousands of files of legacy software files are out there waiting to be used again. You can get to them with the PC/HP-IL interface card if you have it, or using methods like this.
4. Save some time. Imagine typing that entire 16K Startrek Basic program into your 71B, then debugging your typos, only to find out that the whole thing SUCKS. With these procedures, you can take a minute or two to pop it over to your machine and play around with it. Then, when considering the amount of time you have invested in it compared to the amount of time it took somebody to write it, it's not that bad, and actually kind of fun.

Technical Background Information:

Now that we see how useful it can be, we need to have some background information so the rest of it makes sense.

Logical Interchange Format (LIF) Storage Structure and Information:

LIF is the format used to store data on the HP-IL storage devices (tape or disk). A discussion of LIF can be found on-line at ftp://fcext3.external.hp.com/dist/mxd/pc/old/lif_over.html.

LIF storage is fairly simple to understand. The first implementations of this system were on tape, so it makes even more sense. At the start of the tape – or disk– space is set aside on the media for the directory: to record how many files there are, what are their names, where they are located, how long they are, whether they've been deleted or not, etc. After the directory space comes the files stored one right after the other, like on a tape. One thing about LIF, it's entirely one-dimensional: there aren't any of those confusing (or useful) hierarchical directory structures. There's just the files, one right after the other. Of course, LIF must include other details like the volume label, the number of bytes per record, number of records per file, trying to write a new file over the space vacated by a deleted file, etc, etc, but these are not important to this paper. So on the disk (or tape), there's the directory and there's the files.

The Files.

The files as stored are just themselves – ASCII, binary, whatever. If you were to examine the non-directory part of the tape or disk byte by byte, you'd see them all there. And, while you could probably tell where one file stops and another begins, you'd have to be pretty good to tell what each is supposed to do, and you'd never be able to tell what any of them are called (the filename is stored in the directory, NOT with the file). So we all need the Directory.

The LIF Drive Directory.

Here's what part of the directory of one of my disks looks like.

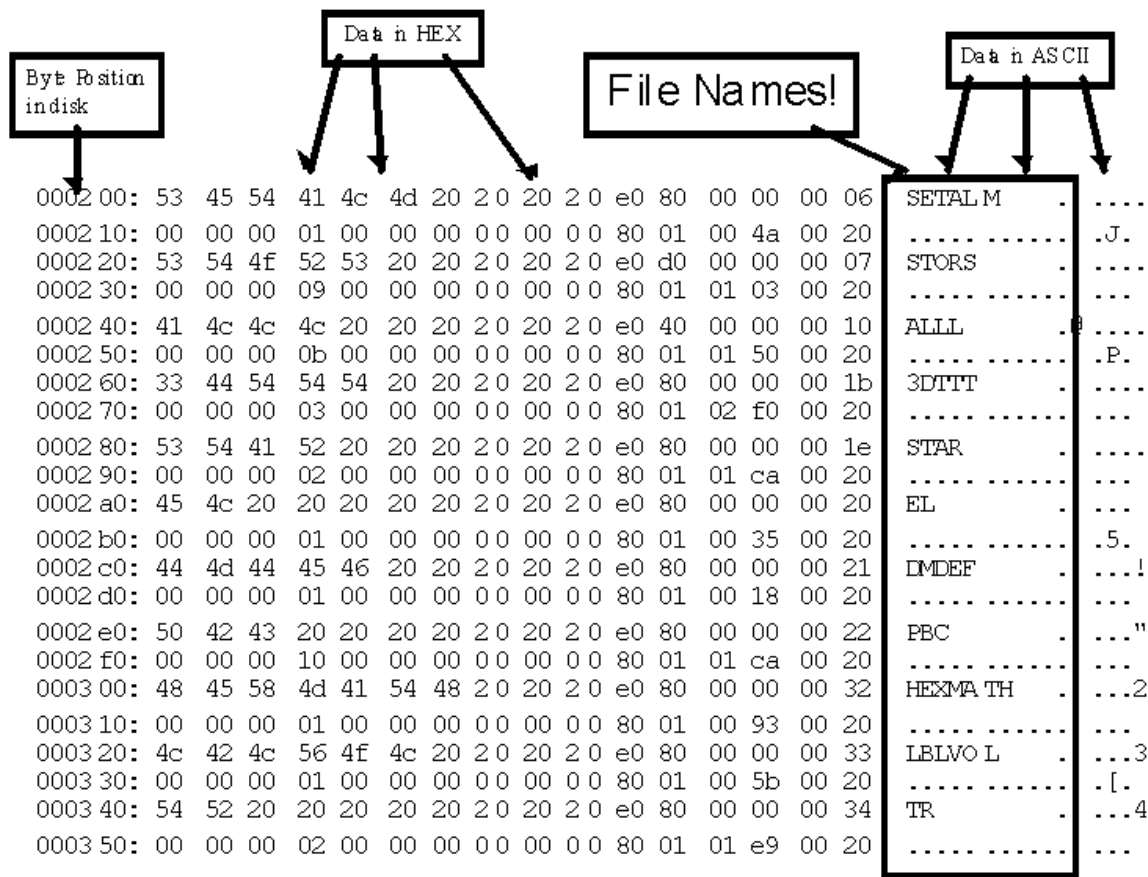


Figure 1: LIF Directory HEX Dump

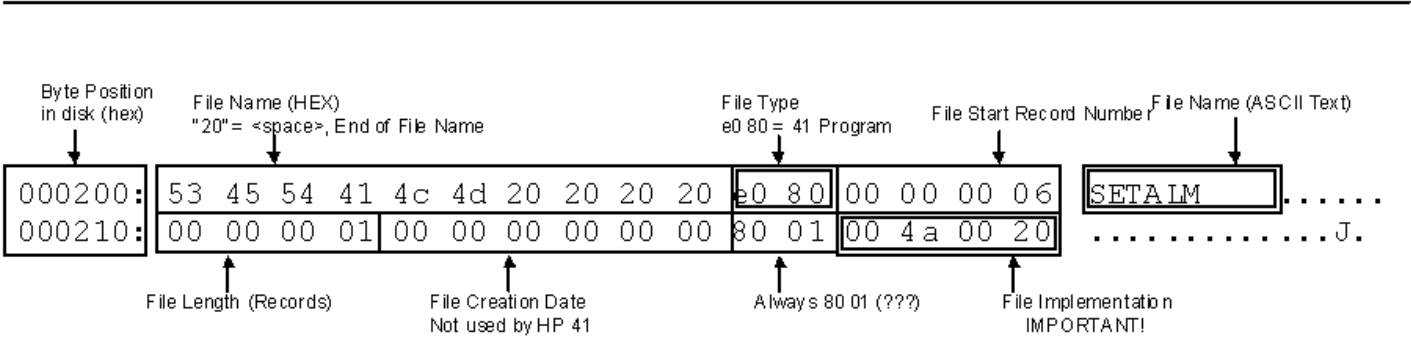
This (Figure 1) is a hex dump of the directory. Hey – we can see what looks like file names in this mess. And, it looks like there’s a file name on every other line. Each file stored on the tape or disk has a directory entry, which takes up 32 bytes of space in the directory area. This is how the size of the directory affects the number of files that can be stored on a disk – if there’s not any more room for a directory entry, then even if space is available for storage, you can’t put any more files on your disk.

The Individual LIF Directory Entry.

Here’s what the directory entry for one file looks like:

```
000200: 53 45 54 41 4c 4d 20 20 20 20 e0 80 00 00 00 06 SETALM .....
000210: 00 00 00 01 00 00 00 00 00 00 80 01 00 4a 00 20 .....J.
```

Individual Directory Entry



Individual Directory Entry - Explained

Figure 2: Individual LIF Directory Entry

So here's what the directory entry for one file contains:

What we care about right now:

It has the File Name (bytes 01-09)

It has the File Type (bytes 0A-0B)

It has the essential File Implementation Bytes (1C-1F)

These implementation bytes (1C-1F Hex, 29-32 Dec) are **very important**. They represent various things about the file (relating to size, file type, number of bytes in file, etc.), but what exactly they represent is not important. What is important is *that the file can not be used if these bytes are in any way different from what they should be*. Programs will not run, data will not be read: the file will be useless. Even though it may have been copied to disk correctly and exist there byte-for-byte identical to its original, if these File Implementation Bytes in the directory are incorrect, the file will not work.

What else the directory entry has:

32 bytes in total

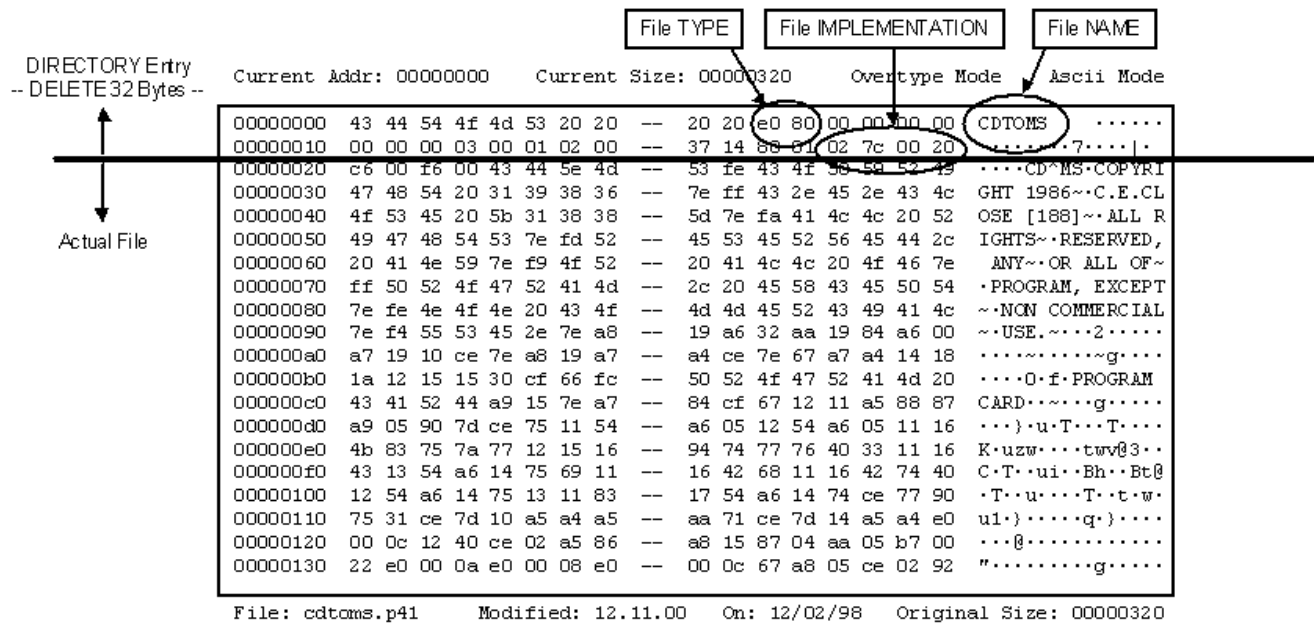
It has the starting record number and file length (in records)

It may have the date/time of file creation (not supported by HP41)

It has the 2-byte value of 80 01 at (1A-1B)

The HP 82973 PC/HP-IL interface Board:

This board was designed by HP to get your PC hooked into the HP-IL loop. Once that was done, you could do a lot of things, including use your PC keyboard instead of those tiny HP-71B keys or store any of your HP-IL calculator files on a DOS disk.



Partial HEX Dump of HP-41 Program File CDTOMS
(DOS File in ftp archive, CHHU06 swap disk)

Warning about the HP82973: It was designed to run at the PC speed of its day (16 MHz), and reportedly *will not run reliably* on today's PC!

HP 82973 file storage:

It was pointed out earlier that having just the file is not enough, you need the directory entry (and the File Implementation Bytes), too. When the HP 82973 board copies a file from HP-IL to DOS, it figures out the directory entry for the file, tacks that info onto the front of the file itself, then stores them together in one DOS file. Presumably, it separates them again when you bring the file back from DOS and puts it where it belongs. I wouldn't know, as I have not used one of these boards.

HP LIF2DOS Software

As time has shown, LIF did not take the world by storm, did not become the de facto standard for disk storage format, and "...increasingly became a relic." (Read HP support's version of this at http://fcext3.external.hp.com/dist/mxd/pc/old/lif_over.html). To extend the life of their product, HP worked up a software package that would allow DOS systems to read LIF-formatted floppies. They called it the HP E2080 LIFUTIL which, although now obsolete, they still remember. HP has thankfully "public domained" this software product, and is giving it away. Refer to <http://www.tmo.hp.com/tmo/support/English/OSKR-OTS:9991:020043.html> for HP's word. There will be a link there to follow for your own LIFUTIL software. Download a copy of this software, and set it up so that it can be run on your computer. *This is the software that will be used to copy files between DOS and HP-IL!*

This is a super software product that will allow you to read and write to a LIF disk from your DOS computer. Despite what HP says about it, it has worked fine for me on Intel 486 and Pentium Processors.

Something Useful: Get and Run a file from the Swap Disk Archive

So now with a little bit of knowledge, let's do something useful! Take a look at the swap file archives online at <http://ftp.math.jyu.fi/pub/hpil/> . Here,

some kind person has taken the years of swap disks from "the old days," converted them to DOS format (using the HP 83973 board) and placed them on-line for anybody to look at or download. Hey, that's great! There's not just HP-41 programs there, but HP71 and 75 BASIC programs, 100's of HP71B LEX files, and a bunch of other stuff. This is the true stuff from the glory days of HP calculators, programs and routines from the calculator crowd at their peak... so how do we get to them?

Don't download them straight from the internet onto your 9114B disk drive and expect them to run...they won't. No! DO download them from the internet to your hard drive. Then, use your little bit of knowledge and spend some time with the file of your choice. Take a binary editor if you have to, and check out the first 32 bytes of the archived file... why it looks like a LIF directory entry! See Figure 3 - HEX Dump of HP41 Program File (DOS ftp file) for an example of what you should see in these ftp files. Write down the things you need (File Type, File Implementation, File Name) then get rid of those 32 bytes and save the file. NOW, take this truncated file and pop it over to your 9114B disk drive. Refer to Appendix I – Basic LIFUTIL Instructions for help copying files using LIFUTIL, you have to use the special user-defined option for file type conversion.

Voila! Now the file is on HP-IL and ready to be run!

Recap and Summary of Steps to Take: (It is easier than it appears, try it yourself)

- 1. Download the file of interest to a local drive
- 2. Binary edit the file
 - a. Record the Important Info
 - b. Delete extra 32 bytes at start of file
 - c. Save the file back to disk
- 3. Copy the edited/shortened file to the LIF disk.
 - a. Follow the guidelines in Appendix I – Basic LIFUTIL Instructions
- 4. Copy the file from the LIF disk to your calculator
- 5. Run the file.

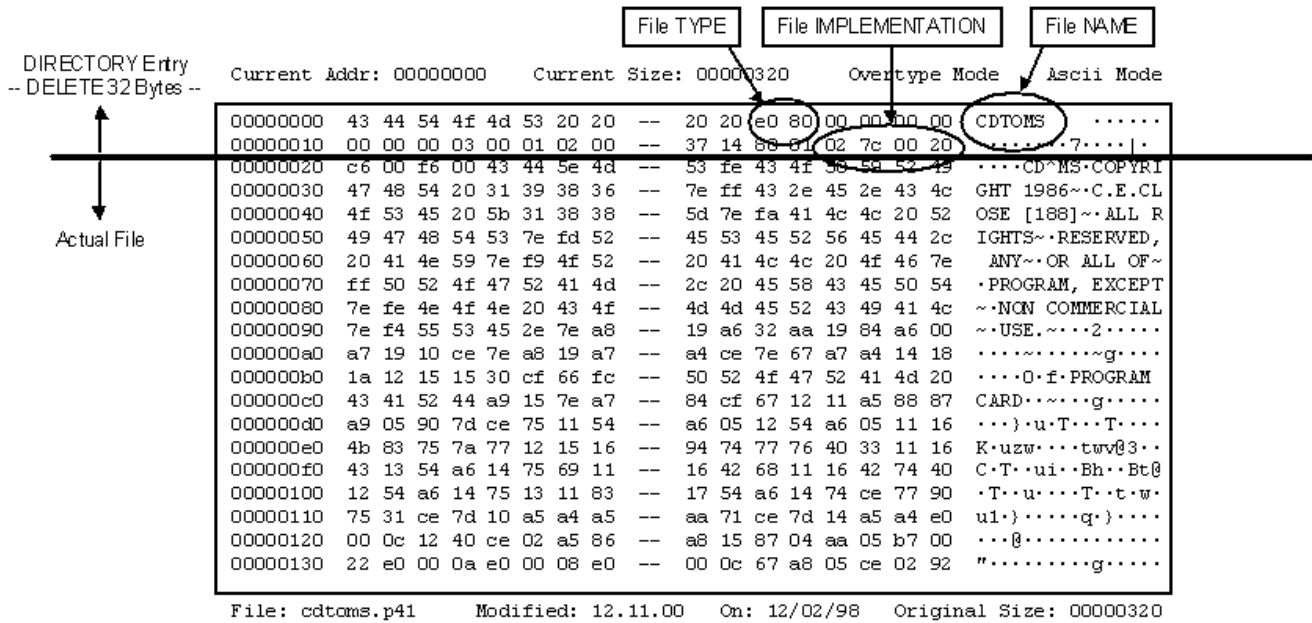


Figure 3 - HEX Dump of HP41 Program File (DOS ftp file)

You may be thinking, "hey, that's really super but there's 1,000's of files in that archive, and I want them all NOW! To open each one, write down the good bytes, then to delete 32 bytes, save, and copy... well... that's WORK!" At least that's what I thought. So I wrote a routine in Visual Basic to open the files, make a record of the file type and implementation, and delete those 32 extra bytes AUTOMATICALLY! AND, I found out another LIF to DOS routine from HP (LIF.EXE) that can be called from a DOS Prompt Command line, so it can be run from a Batch File (remember those?). Check out Appendix II – Automatic Extraction of Important File Info for a summary of instructions on how to do this. Follow these instructions, and you can retrieve files like crazy from the old-time swap disk program!

Something Else Useful: Distribute Your Own LIF Files to Friends and Enthusiasts

Swap disk files from way back running on my machine now! Way cool! But what if I have my own file that I want to distribute? Bar code is still so slow and cumbersome! My 41's batteries don't have the juice left to record cards! What do I do? Well... How about using DOS? How about using email? How about using the internet? All the recipient needs is a 9114B disk drive and some knowledge.

Here's what you have to do: (Refer also to Appendix I – Basic LIFUTIL Instructions)

1. Put your file onto a LIF Disk.
2. Copy it to DOS, using LIFUTIL or LIF. Leave the file type "as-is." This will be the file only, without any of the directory entry information.
3. Find your file's entry in your LIF disk directory. It's OK to use LIFUTIL's hex dump utility for this. Just dump the first 10 or so blocks to the screen, and page through the directory entries until you come to the one you're looking for.
4. Make a note of the File Type and Implementation.
5. Include this information in your email or web page so the recipients can get it to their systems correctly. Since you didn't add the 32 bytes to the file, they don't have to delete them. They do have to know the File Type and File Implementation bytes, though.

So That's That

Now we know how to access/use files that have been DOS-ified by the 82973A board, even if we don't have one. And we know how to DOS-ify HP-IL files on our own, and what information to include with them so they can be used later on by ourselves or others – given that we have HP-IL disk drives.

Important Information:

If you have the HP-IL/PC Interface Board HP 82973, you can ignore most of this.

If you love creating bar code or magnetic cards, you can ignore some of this.

If you don't have a 9114B Disk Drive, keep looking for one.

If you get a 9114B Disk Drive, it will need a new battery, located inside the plastic battery holder. Get a direct replacement from Digi-key for \$16.36 (Digi-Key Part No. P262-ND, Panasonic Part No. LC-R062R4PU).

Useful Information, or "What do Those Files DO, Anyway?"

Pulling out files at random from the ftp archive can be fun but frustrating. Some, but by no means all, of the files include a .TXT file that tells what it does or have text information inside them that explains what they are. For the rest, how does one tell what a file does just from its title? You could load it onto a calculator and print out a listing, but that's work, too. I've written a 41C program "decompiler" that will take a DOS-ified 41C file and create a text source code listing. Special fonts have been created to address the classic "how do you print a 'sigma' or an 'append' character" problem. The other classic problem still exists: how to print out the correct function name instead of XROM(xx,yy). This problem is being solved, but needs more time. Ask me for more information on this program which is still being developed.

A word or two on Formatting LIF Disks:

Disks formatted with the HP-41:

The HP-41 does not recognize or require disk volume labels, so it does not create them when formatting a disk. The problem is that LIFUTIL does require them. Since LIFUTIL expects to see a volume label, it will not acknowledge a disk formatted by the HP-41/9114B combination. I have modified an old program to allow the HP-41 to add a volume label to an already-formatted disk. It requires the Extended I/O module, and details are available upon request. It's easier to format disks with the HP-71 (if you have one) and put a label onto it right away.

There are some limitations regarding the maximum space or number of files that the HP-41 can address on one disk. This seemed to cause some consternation (or at least some programming challenges) in the days of yore, as people wanted to get the most storage possible out of each and every floppy. With today's diskette prices, without a growing number of HP-IL files, and with the capability of using DOS storage devices for mass storage, I am choosing to view this limitation as a non-problem. Solutions and workarounds of the day are available in the journals, if one is really concerned about it.

Formatting disks with the PC and LIFUTIL:

I have not had much success formatting disks on the PC. The program seems to stop with an error before formatting is completed. Others have not had this problem, so try your luck and see what happens.

A word or two on acquiring hardware:

Keep your eye on your local junk sources and government surplus auctions. Buy what you find, keep what you need, sell the extras!

As of summer, 1999, Jim Carter of Interfab had available some HP41/71/75 accessories leftover when Educalc went out of business around December 1997. Most applicable, he had quantities of the HP41/HP-IL Interfaces (HP 82160A) available for \$8.95 each. Call Jim at 949-582-2631.

I monitored ebay for a couple of months (summer 1999), just to see what things were going for at the time. The results are shown in Appendix IV: HP-IL Ebay Prices, Summer 1999 for your information.

References and Resources:

Issue	Title	Author
PPC – LIF Storage Info (partial listing)		
V9 N4 p42-44	Inside the Cassette Tape	Marty Backe (7703)
V9 N6 p3	(letter) –short correction	Bruce Bailey (7115)
V9 N6 p4	(letter) –short correction	Marty Backe (7703)
V10 N6 p28-29	Inside the Cassette Tape – Revisited	Tom Cadwallader (3502)
V10 N10 p9-11	HP82161A Cassette Utilities – DIRectory Backup	Cary Enoch Reinstein (2046)
V11 N9 p55-56	TFSIZE – A Tape Data File Size Finder	Larry Lavins (7310)
V12 N1 p6-8	Sizing and Reading Interchange Files	John S. Chipman (8801)
V12 N1 p8-13	"Disco Skwid" – Disk Drive Limitations and workarounds	Skwid
V12 N2 p15-17	Cassette Drive Problem	Turner Morgan [283], (5766)
CHHU – LIF Storage Info		
V1 N3 p5	Logical Interchange Files	Michael Markov [3]
V1 N3 p12-14	HP-41 Mass Storage Utilities	Michael Markov [3]
V2 N1 p28-29	HP-41 Mass Storage Utilities Part II	Michael Markov [3]
V2 N3 p62-64	HP-41 Mass Storage Utilities Part III	Michael Markov [3]
Swap Disks		
Swap07	Lifdata.txt (LIF Directory Entry Info)	Michael Markov [3]
The Internet		
http://www.hpmuseum.org/	"HP Forum" discussion list, HP Calculator Museum, HP Calculator Classified Ads	
ftp://ftp.math.jyu.fi/pub/hpil/	Archive of HP-IL and swap disk programs and files	
ftp://fcext3.external.hp.com/dist/mxd/pc/index.html	HP's freeware of LIF 2 DOS Utilities	

Acknowledgements and Special Thanks to:

Michael Markov –

for continuing the Swap Disk program, and for particularly useful articles in the journals.

Hewlett-Packard –

for the LIFUTIL program, for "public-domaining" the LIFUTIL and LIF.EXE program, and for the obvious...

Wlodek Mier-Jedrzejowicz –

for his tremendously useful and resourceful book, Extend Your HP41, as well as continued support and encouragement for off-beat little projects such as this.

Richard Nelson –

for his organizational efforts over the years, especially the editing/publishing of the club journals.

Jake Schwartz –

for the PPC CC-ROM project, which made the years of journals available to all.

???? –

for starting and maintaining the ftp archives of HP-IL/LIF/Swap disk files.

My wife –

for tolerating, if not necessarily understanding, activities like this!

Appendix I – Basic LIFUTIL Instructions

Most of the LIFUTIL instructions are self-explanatory or at least easy to understand. When it comes to copying files from DOS to LIF, however, some additional information is helpful. Because LIFUTIL was not written with these HP-IL files specifically in mind, some extra effort is needed to transfer them correctly.

To Copy from DOS to LIF:

1. Select "DOS to LIF Copy" Option
2. Fill in the correct information as requested (file source, destination, etc.)
3. *** IMPORTANT *** for "File format conversion:"
 - a. Press F1 and/or F2 until this option appears:
"Binary image: No format change, user-def. Type"
 - b. Then press F6 to get the special input box that says:
" User-defined LIF file type: E94B
Implementation specific field: hhhhhhhh"
 - c. Enter the correct FILE TYPE (e080 for HP41 programs), press RETURN
 - d. Enter the correct FILE IMPLEMENTATION (viewed with a hex editor), press RETURN
4. Press F5 to start copying.
5. Other, minor yet important notes:
 - a. File names are CASE sensitive, UPPER case is BEST!
 - b. "Fn" keys are instrumental when using LIFUTIL
 - i. F1 and F2 are used to cycle through pre-determined choice of options
 - ii. F3 and F4 (also arrow keys) are used to move between fields
 - iii. F5 usually starts something
 - iv. F7 returns a field to its default option
 - v. F8 usually returns to a previous menu
 - c. Safest to keep "Append to end of disk" for the last option while copying DOS to LIF

To Copy from LIF to DOS:

1. From the main menu, Select "LIF to DOS Copy" Option
2. Fill in the correct information as requested (file source, destination, etc.)
3. For "File format conversion:"
 - a. Use this option: "Binary image: No format change (all file types)"
This is the default option (F7), or press F1 and/or F2 until it appears.
4. Press F5 to start copying.

Special Considerations for TEXT files:

1. DOS to LIF

- a. Select File format conversion: "DOS text as HP BASIC ASCII"
 - b. Entry of File Type and File Implementation information is not required
2. LIF to DOS
- a. Select File format conversion: "DOS text (ASCII, HP-UX, S80 DATA, Pascal TEXT)"

Appendix II – Automatic Extraction of Important File Info

Files you will need:

Hpilhead.xls – this Microsoft Excel file has a Visual Basic Macro which will read from a source file (with extra 32-byte "header" info) and write the truncated file to a destination file, while saving the file name, file type, and file implementation bytes into a text file for later use.

Lif.exe - Another great public domained software program from HP. This one allows DOS command-prompt execution for copying files between DOS and LIF. This program supports user-specified file types and file implementation bytes. These features can be combined to make a DOS batch file (.BAT) that can handle fast copying of multiple files at once.

How to do it:

1. Collect and gather files from the ftp archive that you wish to copy to LIF.
2. Create a listing of these files (including their directory path) in a column in a Microsoft Excel spreadsheet.
3. In the adjacent column to the right, create a listing of where you would like the truncated versions (without the LIF directory info as the first 32 bytes of the file) to be created.
4. Run the Microsoft Visual Basic macro contained in the HPILHEAD.XLS File
 - a. You have to have the current cell be the topmost one in the SOURCE file column
 - b. You have to specify the number of files to be examined inside the VB source code
 - c. Feel free to make a better user interface for this code! I don't have the time right now to do it.
 - d. Bear in mind that a text file with the file names, file types, and file implementation bytes will be created in the default directory. Feel free to make a better user interface for this, too!
5. Use the information in the newly created text file to create a DOS batch file with lines of the following format:

LIF CP <source> <dest> /R /T:0x<bbnn> /I:0x<aabbccdd>

where:

- a. <source> = the file without the 32-byte LIF directory info in it. Include the path.
 - b. <dest> = the LIF floppy drive you are copying stuff to (example -- A:)
 - c. /R = for "raw" option. Otherwise you will get extra bits in the file that will make it not work. Trust me on this, or try it yourself and see!
 - d. 0x = Number-zero, letter-x (Not LETTER-o, letter-x). This tells the program that the following numbers are in HEX.
 - e. <bbnn> = File Type bytes. Include leading and trailing zeros, if any.
 - f. <aabbccdd> = File Implementation bytes. Include leading and trailing zeros, if any.
6. Run the batch file you created, and watch the files fly back to LIF!

Appendix III: LIF File Types

This table shows a listing of all the LIF File Types that I could find information on. These come from various locations in the journals and the ftp archive. I have not verified them all, as I have not seen some of the file types.

71 Data	E0 F0
71 BIN	E2 04
71 LEX	E2 08
71 KEY	E2 0C
71 BASIC	E2 14
71 FORTH	E2 18
71 ROM	E2 1C

File is Purged	00 00
ASCII/TEXT (HP41 AS, HP75 I (LIF), HP71 TEXT)	00 01
Unused, or End of Directory	FF FF
HP41 ROM (ERAMCO) or Extended Memory Files (PANAME)	E0 70
HP41 WALL	E0 40
HP41 KEYS	E0 50
HP75 Text and Key Assignment	E0 52
HP75 Appointments	E0 53
HP75 Mass Storage Data base files	E0 58
HP41 Status	E0 60
HP41 Programs	E0 80
HP75 (P) BASIC Files	E0 88
HP75 LEX Files	E0 89
Visicalc Worksheets	E0 8A
PMS ROM Files	E0 8B
HP41 Data, HP71 SDATA	E0 D0
HP71 Secure Text	E0 D1

Appendix IV: HP-IL Ebay Prices, Summer 1999

Table 3 - Individual Ebay Item Prices

82973 HP-IL/PC Interface	121833884	\$159.49
	128412244	\$51.00
	125128684	\$159.49
	118017680	\$208.51
	---	\$103.50
82160A HP-41 HP-IL Interface	132761340	\$31.00
	11866153	\$40.99
	111685256	\$20.50
	115351333	\$38.00
	119551017	\$20.00
	121221637	\$11.25

	127418052	\$31.00
	127760393	\$7.50
(with CTW book)	135799453	\$41.00
(with 9114B drive)	125735914	\$66.42
9114 Drive Only	128924492	\$51.00
	134990016	
82164A HP-IL/RS232 Interface	113498321	\$72.99
	131563951	\$178.50
	127274796	\$135.50
Series 80 HP-IL Interface	121163184	\$21.00
82169A HP-IL/HP-IB Interface	127285752	\$73.00
	134917689	\$28.72
	132445017	\$46.00
	132634597	\$41.00
	136334922	\$32.00
	139048486	
HP 3468B HP-IL Multimeter	112370258	\$219.89
CMT 128K Ram Disk + 82160A	127247656	\$59.51
82162A HP-IL Thermal Printer	127253522	\$25.00
2225B HP-IL Thinkjet Printer	128928162	\$6.00
82165A HP-IL/GPIO Interface	133733044	\$179.00

Table 4 - Summary of Ebay Prices

82973 HP-IL/PC Interface	5	\$51.00	\$208.51	\$159.49
82160A HP-41 HP-IL Interface	8	\$7.50	\$40.99	\$25.75
(with CTW book)	1	\$41.00	\$41.00	\$41.00
(with 9114B drive)	1	\$66.42	\$66.42	\$66.42
9114 Drive Only	2	\$26.00	\$51.00	\$38.50
82164A HP-IL/RS232 Interface	3	\$72.99	\$178.50	\$135.50
Series 80 HP-IL Interface	1	\$21.00	\$21.00	\$21.00
82169A HP-IL/HP-IB Interface	6	\$28.72	\$73.00	\$36.50
HP 3468B HP-IL Multimeter	1	\$219.89	\$219.89	\$219.89
CMT 128K Ram Disk + 82160A	1	\$59.51	\$59.51	\$59.51
82162A HP-IL Thermal Printer	1	\$25.00	\$25.00	\$25.00
2225B HP-IL Thinkjet Printer	1	\$6.00	\$6.00	\$6.00

82165A HP-IL/GPIO Interface	1	\$179.00	\$179.00	\$179.00
82164A(RS232)/82169A(HPIB)/82143A(Printer)	1	\$185.06	\$185.06	\$185.06

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

Pioneer "Observational Internals"

Posted by [Paul J. Brogger](#) on 2 Apr 2000, 9:29 p.m.

I've found the Pioneers relatively easy to open, repair, and put back together. I thought I'd share my experiences with any who might be interested in trying themselves.

Opening the Pioneer Outer Case

After removing the battery cover and batteries, I use a 1/8" drill bit to cut off the tops of the four heat stakes. (I mount a sharp bit in a counter-sink tool, and rotate it slowly by hand -- one doesn't need to go too deep.) This frees the front and back case halves at the top (display) end of the calculator.

To fully separate the halves, I've then just carefully but forcibly pried the four lower heat stakes apart, working from one side to the other, with nothing other than my fingers. The keyboard is backed by a metal plate and remains relatively stiff, the PCB is free and unstressed, and the back attachment is plastic only.

At this point, you'll have the back cover and front (keyboard) separated. In the older models, the back is lined with an aluminum ground shield, presumably for the reduction of RF interference. In the models with beepers (17B, 27S, & 42S), you'll also see a piezo disc attached to the plastic back.

PCB Removal

If you wish to remove the PCB (printed circuit board) from the front assembly, it comes off easily. Twist the six hold-downs in the obvious directions (back to their original form) and straighten them so they lose purchase and pass through the slots on the PCB. Work with them until the PCB is easily lifted from the LCD/keyboard assembly.

LCD Removal

If you want to go further, you'll have to contend with some nasty glue that attaches the LCD (liquid crystal display) along its edges to the steel frame which serves as the front assembly's "backbone". Work VERY slowly and gently here. I've started prying very gently on one end with a padded screwdriver (a Popsicle stick might be better), slowly but surely separating the LCD from the front assembly. Gentle pressure will free the LCD without breakage, as the rubbery glue gradually gives way from one side to the other -- you'll hear the glue separating as you go. The whole process should be dragged out over a couple of minutes -- don't rush it!

The LCD "zebra stripes" (soft, rubbery connection strips along the top and bottom of the LCD) should be kept clean, as should all of the interior contacts, or they won't work when you put everything back together.

Keyboard Disassembly

If you want to see the keyboard innards, you may be kissing your calculator goodbye, but not necessarily. There are over 50 plastic posts which have been "mushroomed" (melted flat to create a rivet-like head) to hold together the keys, the springy contact sheet, the flexible (mylar?) keyboard circuit, and the metal "backbone". If you clip all these "rivet

heads" off, it will be difficult (but maybe not impossible) to reassemble the keyboard.

I took one apart to see what's inside, but the only reasons I can think of for actual repair are: to clean out some sticky goo; or to replace worn plastic and mylar contact sheets.

Re-assembly

With the LCD and zebra stripes in place, push the PCB slots over the six hold-downs, and use pliers to re-twist them. (Some are used for ground connections too, so make sure adequate contact is made with the PCB pads where appropriate. And don't scratch other traces with your pliers!)

I shave some of the excess "mushroomed" plastic from the four lower heat stakes to simplify re-insertion, and pop the cases back together. I can remove the back any time I want. I suppose that, if dropped, the back will come loose or completely off. However, the PCB / LCD / keyboard bond seems as strong as ever.

Miscellaneous Observations

Two Bezel Types (and incompatible parts)

HP has used two externally different (and readily identifiable) versions of the Pioneer over the years. The early calculators had a flat, two-step bezel around the LCD display. (See the MoHPC image of the HP-32S.) This featured an extra clear plastic (glare-reducing?) layer over the actual glass LCD. On well-used units, dust will often collect between the plastic layer and the LCD -- a very annoying trait, and hard to remedy.

On newer models, the border around the display is angled inward. (See the MoHPC picture of the HP-42S.) There is no plastic layer, and the LCD is itself directly exposed.

It is my experience that later-model displays won't work directly in early-model ("stepped bezel") calculator frames. The space between the bezel front and the metal "seats" for the PCB seems more generous in the older model, and the six metal hold-downs don't seem to be able to create a firm enough bond to make adequate contact between the PCB and the newer display. (I have made one work, by padding the display in front to make it "thicker" and therefore providing the contact needed.) This indicates that there are different versions of the displays, and that one is thicker than the other. (Though I couldn't discern any such difference upon examination -- maybe I'm missing something!)

Different LCD's

There are at least four different LCD units employed in the various Pioneer models, and (as noted above) perhaps a couple variations of some of those.

The high-end models (17B, 27S and 42S) apparently use the same 16x131 dot-matrix display. I'm guessing that the HP-17BII does as well, but I haven't had any of the 17B's apart, and don't know for sure. (If this is so, it should be possible to replace a broken HP-42S display by buying a current-model HP-17BII.) This display enables the two-line display, labeled menus and limited graphing capabilities.

The mid-range models (14B, 22S, 32S, etc.) seem to use the same single-line, large-numeral, alphanumeric LCD. This display utilizes 5x7 LCD dot matrices for more readable characters and numerals, but it is not a general-purpose graphic display.

The HP-32SII display is a variation on the 32S, with even larger, more readable characters and two shift annunciators.

The low-end Pioneer display is a 7-segment display used in the 20S and 10B calculators.

Keys and Keyboards

The Pioneer calculator keys and keyboard are molded in one piece, as part of the front of the calculator. Plastic of contrasting color is injected (or whatever) from the back to produce essentially indelible legends on the keycaps. Each

key attaches to the plastic surrounding it via two thin hinges (again, all part of the same plastic casting). The metal-foil faceplate covers the hinges and the voids which surround the keys and which allow for the keys' movement.

All Pioneer models seem to have the same circuitry implemented within their keyboards. That is, despite their keycap legends, all Pioneer keyboards are functionally identical. It is each different model's particular CPU and ROM logic which implements the varying functionality embossed on the keys. The keyboard is connected to the PCB via a 15-conductor "contact connection", made when the PCB is fastened to the calculator frame.

Circuit Boards

The older Pioneer models utilize a (to me) novel form of surface-mount technology, with rigid plastic frames holding the spidery CPU chip in place via its many leads. Indeed, the CPU is set in a hole punched in the circuit board, and is entirely suspended by its many (VERY many!) hair-fine electrical lines. At first glance, the silvery sheen of these leads looks opaque, but by bringing the PCB close to the eye (or by back-lighting it), one can appreciate that the CPU is "riding on a hammock" in thin air.

The three older USA-built models I've dismantled all had handsome white, translucent circuit boards with anti-corrosion gold plating on the electrical traces. (See the MoHPC internal picture of the HP-32S.) The models from overseas all had green circuit boards, also with gold plating.

The high-end models (at least the 27S and 42S) have longer circuit boards, with spring contacts for the piezo buzzer. They also have had two chips -- the CPU and another 28-pin surface-mount chip, that appears to be an 8K (or so) ROM. There are two rows of pads for another 28-pin chip on each board, but in the several units I've examined, those pads have been empty. There are also several "jumpers" -- pads of interleaved but separated traces, where a small blob of solder will complete an electrical connection. These appear to be associated with write enable and chip select lines. (On one 42S, I tried mounting a static RAM chip, and switched a few jumpers, but never got my "RAM upgrade" to work.)

All the low-end models have had short boards with one chip and a couple of other surface-mount devices. All of the circuit boards appear to be two-sided, with no internal layering.

The one 27S and several 42S boards I've seen differ visually only in that the 27S has a transistor near where the 42S boards have had a diode. The 27S has a clock, so maybe that's what accounts for the difference. Presumably the 27S and 42S ROMs are very different.

With the HP-32SII and newer low-end models, HP has greatly simplified the PCBs' layouts. The new PCBs have traces on only the front side, with a ground shield mesh on the back. (This has eliminated the need for metal shielding in the back of the calculator.) Both the front and back are etched copper, with no gold plating. (The new models may be more susceptible to corrosive moisture, though with the older units' delicate CPU leads so exposed, that may be a moot point.) The CPU chip mount has been simplified: it is still set in a rectangular hole in the PCB, but the chips themselves have more conventional plastic bodies and a surface-mount form.

The new chips' leads have been assigned with simple construction as the goal. As a result, the display and keyboard connections require no crossovers, the circuit board has no feed-through holes drilled, and no jumpers are installed. The new-model calculators could hardly be further simplified, and so are masterpieces of economical design. They lack, though, the nerd-aesthetic appeal of the shiny, more complex older units. (I presume it's the MBA aesthetic that dominates, these days . . .)

Having just obtained my first HP-17BII, I've found that its circuit board and display seem identical to those of the HP-42S. I swapped a 42S circuit board into a 17BII and had a 17BII case that behaved like a 42S. I suspect that the ROMs and the externals (keycap legends, color) are the only differences between the two models. (Which makes the continued availability of the HP-17BII and UNavailability of the 42S even MORE tragic!)

For the time being, 42S fanatics should recognize that one way to insure the longevity of their calculator is to buy a HP-17BII and tuck it away somewhere for future spare parts (especially its display). Also, keep in mind that, until

something like August, 2000 or so, HP may still service (replace!) the HP-42S - it cost me \$69.00 the last time I took advantage of this option.

DO Try this at Home!

If you want some practice before digging into your favorite, or you just want to take a look, you should look around for some used Pioneers with which to fiddle. I found a couple of surplus HP-14B calculators a while back and have made good use of them. Now to me, an algebraic business calculator is just about as useless an item as has ever been created. But at the very least, their LCD's (if not the keyboard components) should be useful for repair of my old 32S. And they came with the nicer, stitched vinyl slipcovers which was a bonus. Finally, the sight of that little CPU chip suspended in space by its external connections is just downright poetic.

Now to me, an algebraic business calculator is just about as useless an item as has ever been created. But at the very least, their LCD's (if not the keyboard components) should be useful for repair of a much-loved 32S. And some come with the nicer, stitched vinyl slipcovers, themselves unused and in mint condition. Finally, the sight of that little CPU chip suspended in space by its external connections is just downright poetic.

Right now, I've got a 42S circuit board and LCD (from a VERY beat-up unit) in a brand-new 14B "body". I've spray-painted the keyboard black (the keycaps were almost all wrong, anyway) and I've memorized the layout. (Besides, you can hold down most 42S keys & see what the functions are . . .) I call it my HP-14BS. A real conversation-starter, and very designer-ish-looking!

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[[Return to the Index](#)]

[[Previous](#) | [Next](#)]

HP-41 Hardware Interface Activities

Posted by [Klug Christoph](#) on 2 Apr 2000, 9:22 p.m.

Also today I like working with the *nostalgic* Fourty-One handheld computer system. From this activities results a collection of HP-41 and HP-IL components. My activities are concentrated to the powerfully CX model, belonging software development includes User code and synthetic programming, not mcode programming. For realizing extensive software solutions I work with RSU (Ram Storage Unit) and burn own ZEPROM-Modules (EPROM).

Main subject is interfacing of external electronic hardware to HP-41 handheld by using HP-IL and HP 82166A IL-Converter. For realizing this I create two systems : First I developed the battery powered, modular Input/Output-Board for use with the IL-Converter. The I/O-Board manual in English language (16 chapters and over 300 pages) contains some analog- and digital-circuits for *measurement and control applications*. Both, HP-41 and I/O-Board creating an expandable software/hardware system for realizing individual applications. The second solution is the IL2000 System. This brings complete I/O-Board technology plus some extended features into a small, handy box, working with surface mounted parts. For the belonging English language IL2000 manual (up to now 3 parts and 100 pages) further updates are planned.

Connecting HP-41 to I/O-Board or IL2000 System make possible data logging applications. For processing measurement-data the transfer to modern PC have a key function for today work with HP-41. By using HP-IL with an own rebuild version of the HP-IL * PC Interface Card I merge mobile HP-41 system and modern PC. Existing transfer software solutions like LINK PLUS or TRANS41 are tested and improved for usage with HP-41.

Above activities completed by writing HP-41 articles as author for the Datafile Journal, published world wide for members of the HPCC Handheld and Portable Computer Club (London).

The collection of HP-41 parts and accessories, HP-IL devices and books and software is an by-product of using HP-41. By trading components with other handheld computer fans I hope to improve existing stock of my collection.

Also for today work the *nostalgic* HP-41 is a powerful handheld computer system : With additionally plug in expansion modules you bypass hardware limitations. And by using HP-IL you realize extensive data storage, printing, plotting, interfacing, measurement and data-transfer to PC.

HP-41 Hardware Interface Activities also includes support of today business of HP-41 users, fans and handheld computer collectors, like trading tools and parts, or exchanging software and books. Please contact :

Christoph Klug Körnerstraße 47 B 31141 Hildesheim Germany

HP-41 Input/Output-Board:

Interfacing extern measurement and control hardware to HP-41 handheld computer using the HP 82166A IL-Converter.

by Christoph Klug

Most people use the HP 82166 A IL-Converter to adapt a standard parallel printer to HP Interface Loop. Alternatively you can use the powerful IL-Converter to connect electronic hardware to the Interface Loop, controlled by the HP-41 handheld computer. I developed an Input/Output Board for the HP 82166 A IL-Converter, designed as a mainboard with four 16 bit extension slots.

HP-41 Interface Loop Controller :

Best HP-41 IL-Controller is the powerful CX model with intern X-Functions, X-Memory and Time-Module. Additionally you need the HP 82160 IL-Module plus EXT I/O-Module to control the Interface Loop. Instead of EXT I/O-Module you can use the Development-Module. For maximum performance you can complete the handheld with the German CCD-Module. Hard- and software of I/O-Board are developed to work optimal in combination with other Loop-Devices like Cassette Drive, Thermal Printer or Video Interface. For that reason you can configurate extensive IL-Systems.

Software for EXT I/O-Module and Development-Module :

For controlling I/O-Board with HP-41 you must alter some internal status register of the HP 82166 A IL-Converter. Therefore you need the EXT I/O-Module or alternatively the Development-Module. For EXT I/O-Module 695 Bytes (= 100 Registers) of software programmed for the HP-41 to control I/O-Board, or 811 Bytes (= 116 Registers) of software for Development-Module.

I/O-Board Software for EXT I/O-Module :

Controlling the I/O-Board with HP-41 CX and EXT I/O-Module is easier and faster than with Development-Module ! The I/O-Module uses the ALPHA-Register to transfer data to IL-Converter. Avoiding problems with loss of leading null bytes in ALPHA-Register, EXT I/O-Module generally works with leading dummy byte D. For controlling the I/O-Board with HP-41 CX and EXT I/O-Module, 24 commands are provided for power up/down, initialisation, addressing, manual service request, interrupt, clear, 16-bit I/O-transfer by ALPHA-Register and by X-Register and 8 bit I/O-transfer by X-Register. For 8 bit I/O-transfer the software provided needs the X-Memory Module for storage.

I/O-Board Software for Development-Module :

Controlling I/O-Board with HP-41 CX and Development-Module is more complex and slower than with I/O-Module ! The Development-Module generates a Buffer to transfer data to IL-Converter. An advantageous feature of the Development-Module is the INTR-routine, which starts HP-41 automatically from an active MSRQ-signal ! For controlling the I/O-Board with HP-41 CX and Development-Module, 22 commands are provided for power up/down, addressing + initialisation, manual service request, interrupt, clear, 16-bit I/O-transfer by X-Register and by ALPHA-Register, 8 bit I/O-transfer by X-Register.

I/O-Board and Time-Module :

You can use Time Module commands for automatic control of I/O-Board. For example power up the system using an alarm function and start some activities like measurement, storing data in main memory or cassette drive, then set new time alarm and put the I/O-Board to sleep waiting for next cycle.

I/O-Board and CCD-Module :

For controlling I/O-Board you do not require the German CCD-Module, but this helpful module expands HP-41 with 100 powerful commands programmed in machine-language. Some are fantastic for use with I/O-Board and make work easy : Synthetic programming, binary-functions for logic bit operations, matrix- functions for data collection applications.

I/O-Board and Zeprom-Module :

The Zeprom-Module is a EPROM housed in a HP-41 plug in modul case, produced by Zengrange (London) who also

produced the well known HP-41 Zenrom-Module. Permanent storing complete I/O-Board control command set inside Zeprom is done by adapting module to small plug in burning converter tool and using corresponding Programmer-Rom software. Using Zeprom-Module, HP-41 main memory is complete free for other program applications and I/O-Board software is protected against memory lost. 51 commands controlling I/O-Board plus 12 utility functions now belonging to CAT 2 and assigned to three User keyboards, which selected by softkeys. Zeprom-Module sum up extension modules and I/O-Board software in only one HP-41 plug in port.

Extern Interrupt :

Inside HP-41 case you can mount a small hardware extension circuit for switching handheld computer from sleep-modus to run-modus by extern interrupt signal. As response to extern event HP-41 starts automatic control of I/O-Board. Because interrupt circuit works with an opto-coupler element there is no risk for damanging HP-41 ! During a running program extern interrupt signal can not switch off handheld computer.

HP-41 and PC :

Normally I/O-Board data processing is done completely by HP-41. But it is possible to transfer HP-41 data-files to a PC. Use a HP-IL * PC Interface Card and Link Plus or Trans41 software. Link Plus only runs on 286/16 MHz PC's and on slow 386SX/25 MHz PC's and emulates two Cassette Drives, a Video Display, a Printer and a DOS file. Trans41 runs on modern PC's and emulates a Printer, a Cassette Drive and a DOS file. Collect data by HP-41 and I/O-Board in the field. Now transfer HP-41 data-files to PC and DOS area. Than import data to EXCEL or other powerful software package for post processing.

I/O-Board Hardware Feature List :

The board uses a basic application for handshake timing from Gary Friedman, described in his book "Control The World With HP-IL". Some more helpful circuits added to the board to improve I/O performance. You find hardware solutions for port addressing, Power On reset, MSRQ (Manual Service Request), Clear function, Power Up/Down function, MSRQ Lock when Power Down and a Reference Voltage. Power supply is external : A special 6V DC-supply unit with optional solar powered rechargeable cell for modular applications. Output voltages are +5V and +15V for digital- and analog circuits. Stand by current of DC-supply (Power Down) is lower than 10 mA !

I/O-Board Port Configuration :

The I/O-Board supports four 16 Bit slots. Every slot is divided into two 8 Bit ports A and B. All slots are connected by a common bus with power supply, handshake lines and data lines. The logic of I/O-Board hardware maximally addresses two 16 Bit input modules and two 16 Bit output modules plugged in at same time ! Address setting is done by jumpers on the modules, not by slot position. Inserted 8 Bit modules need additional software addressing. The slot architecture of the board, hardware and software addressing allows some port configurations for digital and analog input/output.

I/O-Board Plug In Modules :

I/O-Board uses 8 bit or 16 bit add-on modules. On next page is the list of the basic modules for completing motherboard hardware. Furthermore some special modules exist like analog filter modules an RMS measurement module or a computer controlled sine wave generator module. The modular hardware concept of the I/O-Board makes it possible to configurate the interface system with I/O ports to every task you want. You can also develop special modules for your own applications and hardware projects. The I/O-Board is a good basic IL-Device to implement your ideas quickly !

-8 Bit digital input modules and opto isolated input modules. -8 Bit digital output modules, opto isolated output modules and open collector output modules. -8 Bit analog digital converter modules and 8 Bit digital analog converter modules. -12 Bit analog digital converter modules and 12 Bit digital analog converter modules. -16 Bit counter modules. -Quad Multiplexer modules. -Transfer modules for data exchange between two I/O-Boards. -Diagnose modules and Test modules. -16 Bit digital controlled sine-wave generator module. -RMS Converter module (AC

millivolt measurement)

Quality Tests :

Correct and bug free function of software and interface hardware tested with some HP-41 Controllers, some IL-Converters, and eight samples of the I/O-Board which I have built ! Measurement tools and software programs for testing mother board hardware and plug in module circuits exist.

I/O-Board Manual :

The English language manual for I/O-Board (16 chapters and over 300 pages) includes circuit descriptions, circuit diagrams and software documentation for HP-41 CX including barcode plots. The manual is a rich source for users, who want to use the IL-Converter in their own electronic equipment, and for those, who want to learn step by step something about interface technology (read and write bytes to external hardware devices, voltage conversation to the analog area). Test programs included for checking hardware functions. Some practical tips for building, installing and starting with I/O-Board complete the manual, including advanced programming tips and reference lists.

Furthermore manual includes a program for printing HP-41 barcodes on HP DeskJet 500 printer. This needs an HP-41 with HP-IL module, Extended Functions (on a plug-in module or built into an HP-41CX), a CCD-Module plus an HP 82166A IL-Converter to connect the DeskJet 500, or an HP-IL PC interface card to print the barcodes to a printer connected to a PC.

Also manual contains information about HP-41 hardware modifications like building Double X-Memory Modules, interrupt circuit for waking up handheld by extern signal and a extern battery power supply unit. Own chapters describe burning Zeprom-Modules and realizing data transfer or upload from handheld to PC by using the HP-IL*PC Interface Card with Link Plus or Trans41 PC software and improved HP-41 software for transfer main memory register-blocks, X-Memory data- or ASCII files or CCD-Module matrix files to DOS files.

For giving the reader an practical application and imagination about flexibility and amazing performance of HP-41 & HP-IL & I/O-Board one chapter contains the complete hardware and software solution for realizing audio frequency response measurement, including advanced data processing by working with CCD-module matrix files, data storage by X-Memory and Cassette Drive, data output by IL-Printer and IL-Plotter and data transfer to PC !

The manual is completed by a HP-41 & HP-IL reference list, a photo-story about history of developing I/O-Board system and some more photos and details about IL-Devices for measurement, printing&plotting and data-storage. The manual is written for use with I/O-Board, but the manual can be read on it's own for the details and programs it gives. Describing the advanced development, manual continues Gary Friedman's book *Control The World With HP-IL*.

Source of Equipment :

The HP 82166 A IL-Converter is the key part for I/O-Board. Some kits for building this unit including IL-Terminal, IL-Transformer and IL-Chip exist. Printed circuit boards exist for the I/O-Board but presently with only single-sided layout. You must make a lot of additional connections in wire wrapping or soldering technique. That is easy for electronic professionals. For people who have no complex measurement equipment I can align some critical components. Electronic novices can get a completely finished and tested version of I/O-Board including DC Power Supply. Building plug in modules for I/O-Board is not complex and can done by newcomers to electronic construction.

Conclusion :

Today the HP-41 is a nostalgic handheld computer. With added accessories like plug in expansion modules you can bypass hardware limitations of the system. Using the IL-Converter and I/O-Board you open up the interesting field of controlling and developing external hardware. Now you can realize measurement and control from your HP-41, a task normally done with larger systems. Interfacing the handheld to modern PC you can also use modern software tools to analyse the data.

For more information about HP-41, HP 82166A IL-Converter, Input/Output Board and belonging I/O-Board manual write to :

Christoph Klug Körnerstraße 47 B 31141 Hildesheim Germany

I/O-Board Manual Contents :

CHAPTER I Introduction

CHAPTER II HP-41 CONTROLLER

CHAPTER III I/O-Board Hardware

CHAPTER IV DC POWER SUPPLY

CHAPTER V I/O-Board MODULES

CHAPTER VI Practical tips

CHAPTER VII ADVANCED PROGRAMMING

CHAPTER IIX References

CHAPTER IX CENTRONIX PRINTER

CHAPTER X HP-41 BARCODES

CHAPTER XI ZEPROM-MODULE

CHAPTER XII DOUBLE X-MEMORY MODULE

CHAPTER XIII EXTERN HP-41 INTERRUPT

CHAPTER XIV I/O-Board Data Transfer

CHAPTER XV HP-41*PC Data Transfer

CHAPTER XVI AUDIO MEASUREMENT

IL2000 Interface System :

Connecting HP-41 & HP-IL Interface Loop to Future :

by Christoph Klug

When I start 1980 using HP-41 handheld computer, I don't know that this nice machine fascinates me up to 20 years later ! From this work results some special software like a 8 kByte solution for acoustics and loudspeaker development. But most fun I get by interfacing external hardware to the handheld, using the HP 82166A IL-Converter. Some articles about this I/O-Board project are published in former HPCC Datafile Journals. During this two decades I have nice contacts and exchange with other HP-41 enthusiasts and fans world wide !

The existing modular hardware system of I/O-Board (IL-Converter, motherboard, power supply board, lead cell battery, some different plug in modules ...) is ideally for use on workbench, for evaluation and prototyping interface solutions for measurement and control hardware applications with HP-41 handheld computer system. But for mobile and portable applications the I/O-Board system do not have a practical housing like a closed case or a handy box. Furthermore producing of I/O-Board and belonging Power-Supply unit is a mixture of working with printed boards

plus additionally hand wiring much lines and components. To overcome this wake points, now I start the IL2000 project :

New Interface Hardware :

By working with surface mounted components, IL2000 project transfers complete proved I/O-Board technology into a handy box. Some interface- and power supply features expanded : Addressing maximally 8 sixteen bit I/O ports (I/O-Board addresses only 2 sixteen bit I/O ports), a three colour diagnostic LED, some optional supply voltages, a solar cell charger option and a low bat indicator circuit. The new modular mechanical design is realized by 10 inch rack case parts. Some parts are modified for reaching standard height of only 88mm. For extensive interface applications, mounting more hardware modules is possible by enlarging box width up to 19 inch rack size.

IL2000 interface board + IL-Converter + POW2000 power supply board stacked together, forming the basic hardware module, placed to right side of 10 inch rack case. A 50 pin flat ribbon wire power- and data bus is connected to additionally plug in modules : I/O2000 is the adapter solution for employing proved I/O-Board plug in module family. Inserting two 16 Bit or four 8 Bit plug in modules is possible. DIG2000 board make possible general 16 Bit digital I/O applications. Additional evaluation area including a ground-plane gives you some space for placing own hardware circuit extensions.

HP-41 software used for IL2000 system is compatible to I/O-Board. Refer to I/O-Board manual Chapter II and Chapter VII . Addressing additional input- and output ports needs expanded software. Like I/O-Board also IL2000 interface board works with pulsed GETO-Line (IL-Converter trigger signal) for port addressing.

IL2000 system is the multi-purpose interface solution for expanding HP-41 handheld computer system by advanced input/output applications. The portable, battery powered design is helpfully for today HP-41 fans and enthusiasts, and for electronic professionals. The modular hardware concept make possible configuration of IL2000 interface system to every task you want. Developing special modules for you own applications and hardware projects is possible in short time. IL2000 system continues proved I/O-Board design, and both systems are fantastic IL-Devices for implementing your ideas quickly !

IL2000 Manual :

New IL2000 interface system is documented by an own manual. This complements the existing 300 page I/O-Board manual. Like the history of growing I/O-Board manual also for IL2000 manual future updates are possible, describing more hardware applications and system expansions. The first print of IL2000 manual includes some photos, diagrams and more than 100 pages. First chapter contains prefaces, written by the handheld computer book authors Gary Friedman (Control The World with HP-IL), Dr. Tony Duell (HP-48 I2C-Bus Interface) and Wlodek Mier-Jedrzejowicz (Extend your HP-41). Second chapter describes the complete hardware and the mechanical design. Third chapter gives a first example working with IL2000 system : Third octave frequency response audio measurement. This application continues audio measurement chapter XVI of I/O-Board manual. For getting a own sample of the new IL2000 manual contact the author.

For more information about HP-41, HP 82166A IL-Converter, IL2000 System and belonging IL2000 manual write to :

Christoph Klug Körnerstraße 47 B 31141 Hildesheim Germany

IL2000 Manual Contents :

PART I Prefaces from Gary Friedman, Christoph Klug,

Dr. Tony Duell and Wlodek Mier-Jedrzejowicz

PART II Connecting HP-IL Interface Loop to Future

PART III Audio Third Octave Frequency Response Measurement

Audio Measurement with HP-41 :

by Christoph Klug

The HP-IL Interface Specification 82166-90017 describes a scenario for realizing audio measurement with HP-IL : The application use the HP-85 Personal Computer, controlling the HP 3468A IL-Digital Multimeter for measurement of AC voltage levels. Additional HP 82166A IL-Converter is interfaced to a Multiplexer-Device for connecting Digital-Multimeter to different points of measurement object, controlled by software.

Today the modular, battery powered I/O-Board make possible similar measurement application with minimal hardware expense. Chapter XVI of I/O-board manual contains detailed informations : By using modern analog IC's we expand the existing I/O-Board plug in module family by two special modules. The Micro Linear ML2038CP is a one chip sine-wave-generator (internal wave-table-rom and 8 Bit DAC), controlled by parallel 16 Bit data word. And for measuring AC voltage levels we take the SSM2110 RMS-Log-Converter chip (from Analog Devices) as frontend for the existing 12 Bit ADC-Module.

The rest is done by software : For HP-41 we program control routines for setting frequency of sine-wave-generator, and for reading voltage level to X-Register. Working with high accuracy reference measurement equipment during calibration (Rhode & Schwarz Audio Analyzer UPA 3) and using curve fitting routines (HP-41 AEC-Rom) our low cost system reach amazing performance : Generator frequency runs from 5Hz to 60KHz with 1Hz resolution. Measurement of AC voltage levels runs from -80dB (=0,1mVeff) to +20dB (=10Veff), typically accuracy is +0,3dB !

For determining the complete frequency response (for example of a Hifi-Device) we need additional software. By entering start- stop- and increment-frequency we execute an automatic measurement sequence. Voltage level readings may be stored in X-Memory data-files. But much better is working with CCD-Module matrix-files. A measurement sequence creates two matrix-files, one for parameters (name, date, time, start- stop- increment-values ...), belonging second matrix-file containing measurement data (AC voltage levels).

By using CCD-Module math-matrix commands we program a helpfully math-function set for calculating frequency response data : Invers frequency response, adding frequency response data, shifting, finding max peak, smoothing (for acoustical measurement). For completing this audio-measurement system we add some HP-IL Devices : IL-Cass-Drive for extensive data storage, IL-Printer and IL-Plotter for data output. Using HP-IL*PC Interface Card, we transfer measurement data to PC, making possible data post processing with EXCEL.

Part III of IL2000 manual continues chapter XVI of I/O-Board manual and describes additional HP-41 software for carrying out third octave frequency response measurement. Additional software make possible Multi-Plots using IL-Plotter. Furthermore some optional hardware extensions given like adapting a microphone for acoustical measurement or an expanded input multiplexer section with high impedance line level input, 10KW input, 16W input and a balanced input.

For more informations about using HP-41 for Audio Measurement write to :

Christoph Klug Körnerstraße 47 B 31141 Hildesheim Germany

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]

 [Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index\]](#)

[\[Previous | Next \]](#)

Repairing an HP-41CX

Posted by [Stefan Vorkoetter](#) on 2 Apr 2000, 9:20 p.m.

Recently, my HP-42S broke down (the "9" key stopped working) and I found to my horror that there was no way to disassemble the calculator. I e-mailed all my colleagues to see if any of them had any old HP calculators lying around, and one of them did. He gave me his old (broken) HP-19C and HP-41CX.

The 19C was fairly easy to repair. It just needed a new battery pack, and the keyboard had become unplugged from the CPU board. There were also a few other minor repairs to make.

The 41CX was more challenging, but first a bit of history.

Long ago, I had an HP-41CV and one day it stopped working. I found that the CPU board is connected to the keyboard by a set of flexible connectors which rely on pressure to maintain contact. This pressure is provided by the screws that hold the calculator together, but the plastic posts that the screws go into had cracked (interestingly enough, my Braun shaver suffered the same failure on the very same day). A similar arrangement connects the module back-plane and battery connectors to the keyboard circuit. The repair consisted of a few drops of cyanoacrylate glue, and a wrap of fine wire around the posts until the glue had thoroughly set.

Unfortunately, a few months later, I sold the calculator since I thought I might like one of those nifty BASIC-language pocket computers better. I was wrong, so the following year my wife got me an HP-42S for Christmas. It didn't have that same air of quality that the 41CV had, but it worked fine, and I'd been using it for about 10 years now... until the "9" key stopped working.

My colleague's HP-41CX was in need of more extensive repairs than my 41CV had needed. Here's a brief list of what was wrong:

- All four screw posts were cracked to varying degrees.
- One of the three keyboard to CPU board interconnects was missing.
- The battery contacts were badly corroded.
- Some external parts were missing: two rubber feet, the charging jack cover, one module port cover.

On a positive note, the calculator did come with a number of useful things:

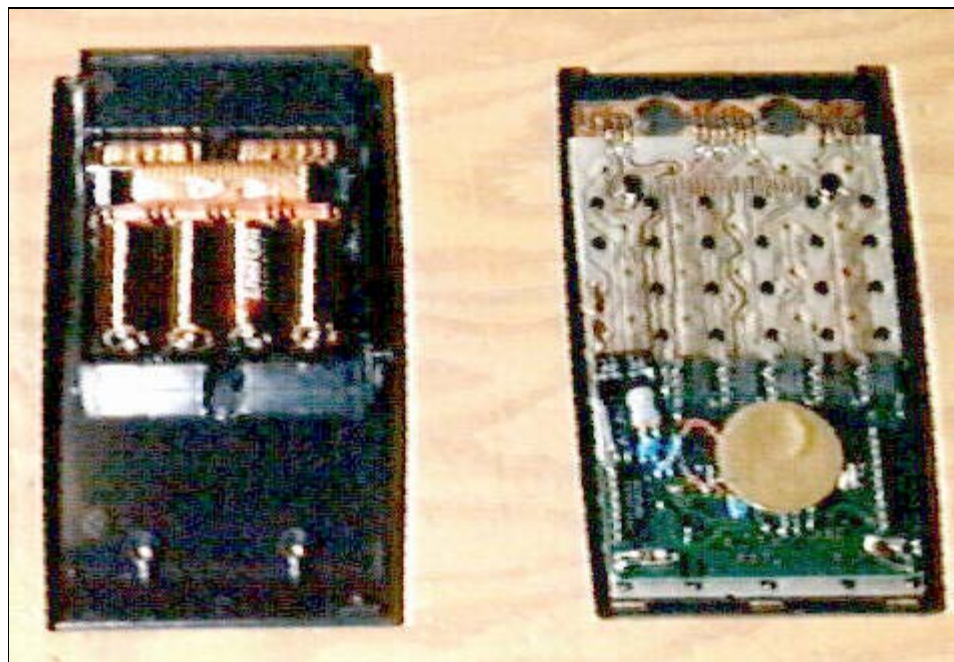
- Volume 2 of the Owner's Manual (which seems to contain all the useful information, so I wonder what's in Volume 1).
- A card reader!
- The manual for the card reader and a card wallet full of cards.
- A package of 120 blank magnetic cards.
- The carrying case.

In this article, I will describe the steps I took to restore the calculator to working order. I will assume you have at least

hobbyist-level skills in electronic equipment repair. I'm not an EE by any stretch of the imagination, but I've been tinkering with electronics since I was a little kid, so I don't mind taking things apart to try and fix them.

This information is provided as-is, with no warranty of any kind. Under no circumstances will Stefan Vorkoetter or the Museum of HP Calculators be liable for any loss or damage resulting from use of this information. If you attempt to repair your calculator using this information, you do so at your own risk.

Inside the HP-41CX



This photo shows the two halves of the HP-41CX case.

On the left is the back of the case. At the top are the module ports. The orange colored thing is the backplane, which connects the keyboard circuit board to the module ports, and also to the battery. Just below this you can see the battery, consisting of four size N alkaline cells. Near the bottom of the back are the holes through which the bottom screws pass.

On the right is the top of the case. At the top is the LCD, and the two dark circles on the back of it are the LCD driver chips. Below that and extending all the way to the bottom is the keyboard, held to the case by black plastic heat stakes. Also at the bottom is the

CPU board, which is connected to the keyboard by flexible interconnects, aligned with the keyboard by the bottom screw posts, and held in place by the back of the calculator. The big yellow circle on the CPU board is the speaker (used by the BEEP and TONE functions, and the alarms).

In addition to the front and back case pieces, there's a 1/4 inch thick bezel which goes between the two case halves (from the outside of the calculator, it's the shiny band between the top and bottom matte finished case halves).

To disassemble the calculator, lay it face down on a soft surface and remove the battery. Peel the rubber feet off the back to expose the screws. Undo the four screws, and lift the back case off of the front. Then lift off the bezel, being careful to note its orientation.

Next, lift off the CPU board, and the black plastic insulator sheet (note its orientation; it has a notch cut out of one side). Removing the CPU board will expose the CPU to keyboard interconnects, and the black plastic frame that holds them in place (this might be slightly different in the 41C and 41CV models).

Repairing Cracked Posts

I suspect this problem must be common to lots of HP-41 series calculators. The screw posts are under a great deal of stress, since they must hold the machine **tightly** together. Any jolts received by the calculator take their toll on the screw posts. Over time, they're bound to weaken and crack.

There are four screw posts in the HP-41, two near the bottom and two just above the battery hatch. These are visible in the photo at right, which is of the front case and keyboard, with the CPU board removed. The two bottom screw posts are in

the second row of black dots from the bottom, and the two top ones are just above the topmost row of black dots (the black dots are heat stakes that hold the keyboard to the front case).

Notice the two rows of contact fingers between the top screw posts, and the two long rows and one short row between the bottom screw posts. (I think the 41C and 41CV don't have the short interconnect.)

Repairing cracked posts is fairly simple. First disassemble the machine and remove the CPU board. Also remove the three flexible interconnects, and the plastic frame that holds them in place (be careful not to crack the frame).

Apply a drop of thin cyanoacrylate adhesive (or liquid styrene solvent) into the cracks, and quickly tightly wrap each post with several turns of fine single-strand wire (I use 30 gauge wire-wrap wire, stripped of its insulation). Twist the ends of the wire to keep it from unwrapping. When the glue has set thoroughly (a few minutes for cyanoacrylate, or overnight for styrene solvent), unwrap the bottom posts. The top posts can remain wrapped since nothing has to fit over them. Apply some cyanoacrylate to the wire around the top posts and trim off any excess wire.

If cracked posts were the only things wrong with your HP-41, you can now reassemble it. Reinstall the frame and flexible interconnects and slide the CPU board back over the posts. Place the black plastic insulator sheet over the keyboard. Put the bezel on the front half of the case (note that the bezel is slightly narrower on one side than the other; the narrow side faces the front of the case). Place the back case onto the bezel, and screw everything back together (the short screws go in the bottom holes). Tighten the screws securely, but do not overtighten them. Reinstall the battery pack and turn it on. If all is now well, the calculator will display "MEMORY LOST".

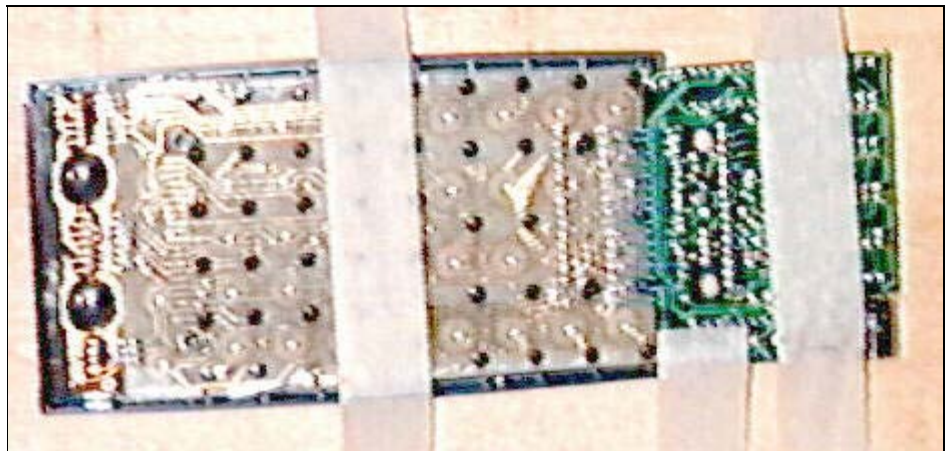
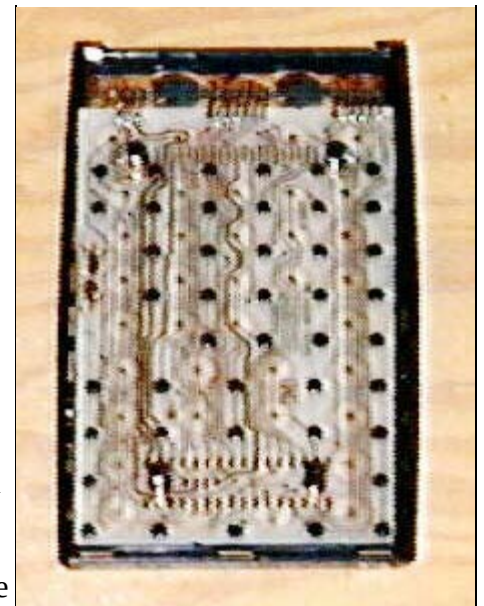
Replacing the CPU-to-Keyboard Interconnects

On my HP-41CX, one of the three flexible CPU board to keyboard interconnects was missing. It was most likely lost during a previous repair attempt (the rubber feet were also missing). I decided to do away with the interconnects entirely, and hardwire the CPU board to the keyboard. This has the advantage that the connection will no longer rely on the integrity of the bottom screw posts.

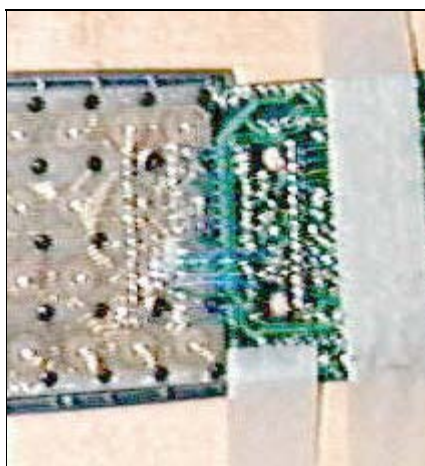
The first step is to mount the calculator to a convenient surface, and then mount the CPU board next to it so that the wiring can be carried out. Here, I've taped them both to a board. Overlap the CPU board on the bottom of the calculator, component side down, as shown. Be sure to orient the board correctly.

Start by tinning all the conacts on the keyboard and CPU board using a fine-tipped soldering iron.

Cut 16 pieces of 30-gauge wirewrap wire to the appropriate length to reach from the bottommost contacts on the keyboard to the corresponding contacts on the



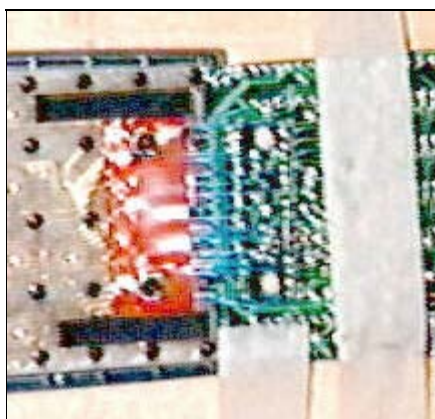
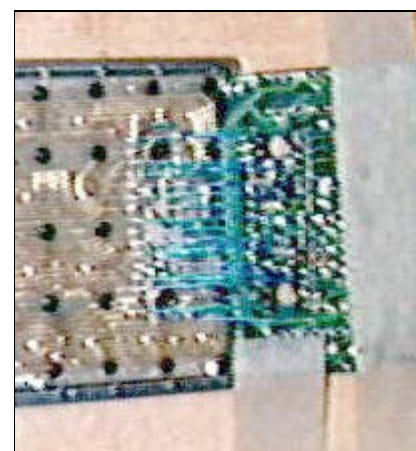
CPU board. Strip 1/8" of insulation from each end of each piece, and solder the 16 wires between the appropriate contacts as shown.



Next, cut 4 pieces of wire of the appropriate length to reach from the middle row of contacts on the keyboard to the corresponding contacts on the CPU board. Strip 1/8" insulation from each end, and solder them in place as shown.

Now it's time to move on to the final row. Cut 14 pieces of the appropriate length to reach from the top row of contacts on the keyboard to the corresponding rows on the CPU board. Strip 1/8" insulation from each end, and install them, omitting the two locations that are blocked by the screw posts.

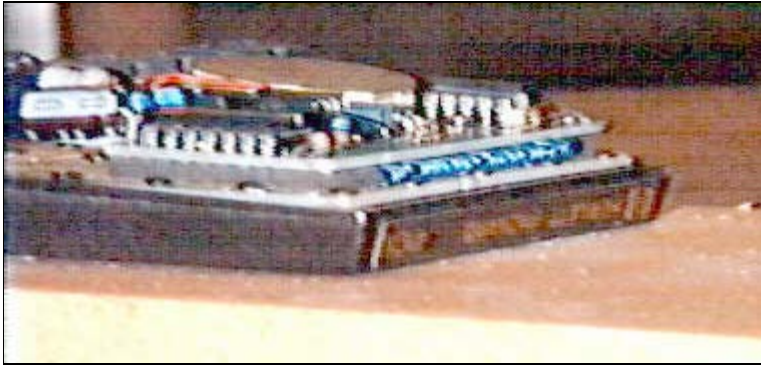
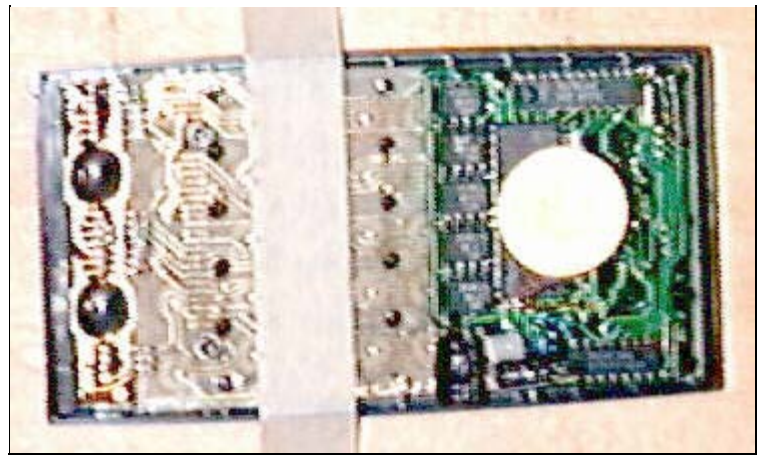
Cut two additional pieces about 1/4" longer to connect the remaining two pairs of pads. Route these wires around the screw posts (and the screw post holes in the CPU board), and solder them in place.



The next step is to reinstall the CPU board on the keyboard. Before doing this, cut a piece of electrical tape to the same length as the CPU interconnect frame that you removed, and punch two holes in it to clear the screw posts. Also glue some 1/8" thick foam rubber to the keyboard circuit board at the two ends of the tape to hold the CPU board away from the keyboard once it's reassembled.

In the photo, the electrical tape is red, and the foam rubber is the two black strips above and below the taped area.

Finally, carefully fold the CPU board onto the keyboard, lining it up with the posts. It helps to lay a narrow metal ruler over the wires at the place where they will bend (half way between the corresponding sets of pads) to get them all to bend in the same place. Push the CPU board down over the screw posts until it is sitting on the foam rubber, parallel to the keyboard. The photo at right shows the CPU board in position. The photo below shows a side view, with all the wires (blue) visible. You can also see the foam rubber strips on which the CPU board is resting.



At this point, if there's nothing else to repair, you can reassemble your HP-41 as described in the previous section. You might have to trim a half inch or so off the bottom of the plastic insulator sheet to clear your foam rubber strips. You may also find it a bit harder to tighten the bottom screws because the wiring and foam padding might provide some resistance, but they will give. Just be careful when tightening not to strip or crack the screw posts.

Cleaning or Replacing Battery Contacts

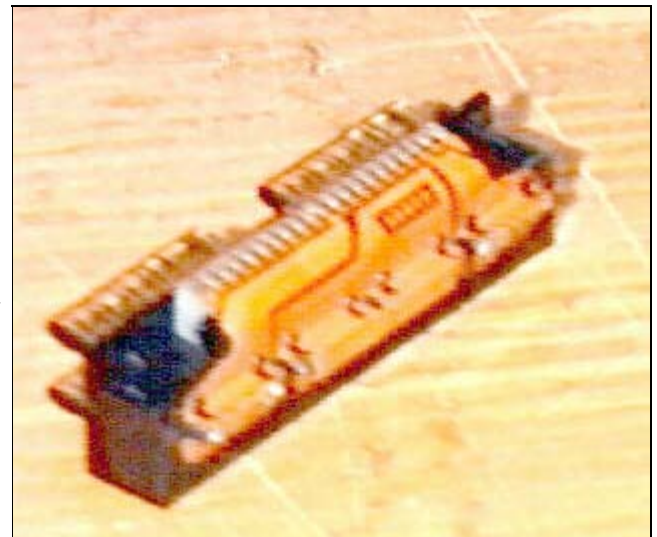
My HP-41CX had badly corroded battery contacts because the last set of cells had been left in it for several years. Sanding the contacts merely got right down to the plastic under the contacts.

There are four battery contacts. When viewed from the back of the calculator, with the module ports at the top, the leftmost contact is the negative one. The rightmost is positive. The two in between aren't connected to anything except each other, and are just used to connect the two middle cells to one another when the battery pack is in place. The battery contacts are part of the module port back-plane, and the whole assembly is just a friction fit into the back of the case. It's fairly easy to pull out to work on it.

I cut out a 1/8" wide strip of thin brass strip (available at any hobby shop) wide enough to span the two middle contacts, and glued it across them with cyanoacrylate. This strip now serves to connect the two middle cells.

For the positive and negative contacts, I was able to expose some uncorroded copper on either side of the center of the contact. I cut two more brass strips, each the width of a contact, and carefully soldered them to the clean copper on either side of center, effectively making a new contact.

The photo at right shows the end result. It's a bit fuzzy, but if you compare it to the back-plane from your calculator, you can see that the middle two battery contacts have been bridged, and that new contacts have been soldered onto the outer two.

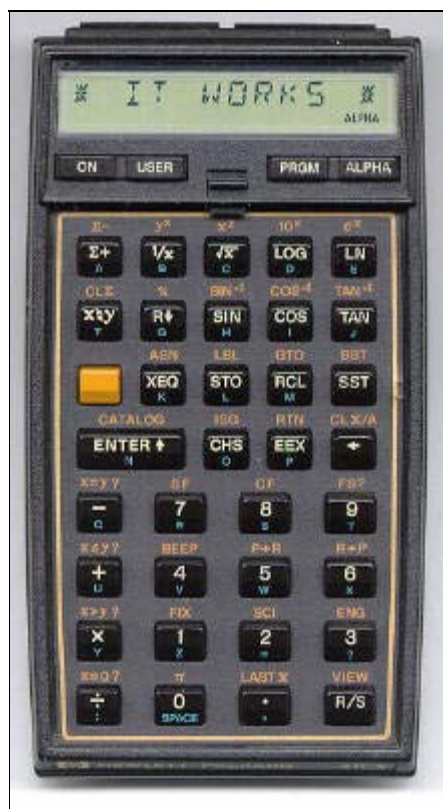


To solder the new contacts in place, first tin the copper areas and apply a small amount of solder. Then tin the back side of the brass strip. Hold the strip in place against one side of a contact, and touch the soldering iron to the front

side of the strip. The iron will heat the strip, and the solder behind it, and fasten it to the tinned copper area. Do the same with the other end of the strip. You have to do all this quickly, or you risk melting the plastic backing of the battery contacts.

When reassembling the calculator, clean the back-plane contacts that connect to the keyboard with some rubbing alcohol. Chances are they will have been contaminated with finger prints and solder flux while you were repairing the battery contacts.

It Works!



After several hours spent repairing the 41CX, I put it all back together, installed the battery, and was greeted by a "MEMORY LOST" message. A quick test revealed that everything was working.

I also tested the card reader that came with the calculator. It seems to work most of the time, although it sometimes makes funny noises and gives an error message while reading. I intend to open it up and see what's wrong in the near future.

I then proceeded to give the machine a thorough cleaning. I used a very slightly dampened rag to clean all the keys, and the spaces between them. After a few minutes, I had a good-as-new HP-41CX. In some ways, it's actually better than new, since it no longer relies on pressure to maintain the critical keyboard to CPU interconnection.

There are still a few things that need to be fixed. The cover for the charging jack is missing (although the calculator didn't come with a rechargeable battery). One of the port covers is missing (the card reader was plugged into it when I got it). The ALPHA key sometimes bounces, acting as if I'd pressed it twice in a row. And there is a strange little pair of bumps on the right side of the keyboard, next to the SST key (see the close-up photo). If anyone can tell me what that might be, [I'd appreciate hearing from you.](#)



Wrapping Up

Well, that's all that I needed to do to get this calculator working again. If you take it a step at a time, it's really not very difficult to do. Just be careful, check all your work as you finish each step, and your chances of success are very high. On the other hand, if you're not comfortable disassembling electronic equipment or working with a soldering iron, you might be better off getting a more experienced friend or colleague to help you out.

If you or someone you know has any HP-41 accessories (especially a module cover, charging jack cover, and keyboard overlays), I might be interested in taking them off your hands if the price is right (i.e. nil to very low). Please [contact me](#).

Finally, if you need any help or advice on repairing your own HP-41, feel free to [e-mail me](#), but please do not be surprised if it takes me a few weeks to get back to you.

Edit Message

Delete Message

Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)

HP Articles Forum

[\[Return to the Index \]](#)

[\[Next \]](#)

Hewlett Packard HP-25 Calculator: The Minimum Computer

Posted by [James Blodgett](#) on 2 Apr 2000, 9:14 p.m.

The Hewlett-Packard HP-25 programmable calculator is the cheapest complete computer system on the market today. This statement may raise debate. For one thing, I am saying that the HP-25 is a computer while several cheaper programmable calculators are not. Second, despite its simplicity I am saying that it is a complete system. Both points are matters of definition and debate, but my definition makes good sense.

What, after all, makes a computer? At what level of complexity does a programmable calculator become a computer? At present common usage gives no precise distinction. But since loops, nested loops, and alternate sequences of operations are fundamental to programming, I would like to suggest that a system is a computer if it can do them, and is not otherwise. In general, this means that it must be able to do some form of conditional branching. Besides, I have tried programmable calculators without conditional branching, and they are simply not much fun. They can do only one short sequence of steps in one way, so the complexity of their programs is strictly limited. Conditional branching allows complexity. And the HP-25 is the cheapest programmable with conditional branching.

(This is not to say that a cheaper programmable might not be the best buy for many people. For some practical applications, a little programming goes a long way. And on some models, it is possible to simulate simple loops by pushing some version of the "run" button over and over.)

Another possible point of contention is my calling eh HP-25 a complete system. I call it complete because it computes, by itself, and it is all you have to buy. At this writing (Feb '76) there are more ambitious computers advertised in the price range of the HP-25, but they are by no means all you have to buy. for example, the Altair 680 CPU board in kit form lists for the same price as the HP-25 at retail (\$195), although the 680 is not yet discounted while the HP-25 is. But the 680 CPU board is a "complete computer" only to those who are ready to wire it into something else--and that something else had better include some memory. Even the 680 "complete" kit with memory, front panel and case for \$345 is a complete computer only to those who are willing the enter and read out both data and machine language instructions in binary, do this each time the machine is turned on, and flip switches forever. This might provide interesting recreational computing for some people, but the machine by itself is certainly not very practical. Most Altair users buy at least a terminal, the Basic language (available as of Feb. '76 only on the Altair 8800) and the extra memory necessary to support Basic, and these additions put the system in the \$2,000 range.

One might also question my statement that the HP-25 is a complete computer because of features which it lacks, or because of the general limits of the system. Hewlett-Packard, for example, calls the HP-25 only a "programmable calculator, " while their HP-65 is "fully programmable" because it can store its programs on small magnetic cards. Such storage is certainly a useful feature, but it is not a necessary feature for programming to intellectually "fell like" computer programming. The HP- 25 feels like a computer, and it doesn't need a card to do so.

(The new Texas Instruments SR-52 also has a card, and twice the memory and program capacity of the HP-65 for half the price. Readers contemplating purchase might well consider the SR-52; the HP-25 is the cheapest but not necessarily the best choice.)

The HP-25 is in fact quite limited for a computer: it has 49 steps of programming and 8 memory registers, plus 5

registers in its operational stack. Refusing to call it a computer because of its limits, however, is mainly a matter of preference for larger systems. I would call the HP-25 a computer despite its limits. The system it defines is not at all trivial. It can do enough so that it is fun to program, and it is also useful, and will do a reasonable fraction of the personal computing that a mathematically oriented user would do with a much larger system.

Indeed, the very fact that the system is limited has both practical and recreational advantages. A larger system can be very impractical when one is seduced into wasting days solving what had at first appeared to be a simple problem, writing hundreds of steps, debugging, and improving the output. If one has an HP-25 and a problem can't be solved on it, one knows that the problem will take a substantial amount of programming on a larger system, and this is a very good point at which to ask whether the solution is worth the trouble.

Another recreational and educational advantage of a limited system is that one is much more quickly forced to optimize programs. Optimal programs are much better esthetically than sloppy programs, and there is a real feeling of craftsmanship in knowing a system well, trying every trick in the book, and finally being able to shoehorn a complicated problem into a limited number of steps.

An example of a recreational problem may give some idea of the possibilities and limits of the HP-25. I have been trying since I bought the machine to make it play ticktacktoe. I am almost certain that this is impossible in such a limited machine, but it is less impossible than one might think at first.

For example, there are nine positions in a ticktacktoe board--how can one store the contents of these positions in only eight memory registers? Well, the machine can take a number with a decimal point and throw away either the fractional or the integral component. Thus it is possible to extract a specific digit in a larger number by placing the decimal point in front of the digit, taking the fractional component of the number, then multiplying this fractional component by ten and looking at the integral component. It is also possible to place a digit in any decimal position by multiplying the digit by ten-to-the-power of the desired decimal position and adding the resulting number to the number in a storage register. (Unfortunately the ten-to-the-power function is slightly off in the last decimal place for ten-to-the-seventh and higher powers, but if necessary this can be corrected by a few steps of programming.) Since one can both take out and also put a number in any decimal position, the ten decimal places in one register can be used as ten different memory locations. Thus if necessary a representation of the ticktacktoe board can be stored in only one register.

A more fundamental limitation for implementing ticktacktoe is that the small number of branches which are possible in 49 steps seems much lower than the number of decisions a computer must make in playing tacktacktoe. The number of branches possible for the HP-25 is much less than 49 because a branch requires from one to five steps depending on what one is doing. In order to branch it may be necessary to bring the two numbers to be compared into the two appropriate registers, specify the logical test to be applied, and specify the two different directions to go to depending on the outcome of the test, and this totals five steps.

My latest thought is to sequentially peel off digits from a sequence of memory registers and use these digits as a sort of higher-level programming language to specify the sequence of application of a group of subroutines, thus in a sense expanding the programming capacity. I am fairly sure that this will not work, since the peeling-off and branching program alone will take quite a few steps, and there might be room for only about four or five very short subroutines.

Whether or not it can be implemented, the ticktacktoe problem is an example of the complexity and richness of strategy possible in what seems at first to be a very limited system. And despite the difficulty with ticktacktoe, by no means are all games excluded. For example, the Applications Programs book that comes with the HP-25 includes a moon-landing program and a version of nim. And I have written an ESP testing game, a ball-bouncing game, and others. Games which can be implemented tend to be relatively simple, however. The best game is the system itself.

From the Nov-Dec 1976 issue of Creative Computing: A Creative Computing Equipment Profile. Reprinted in The Best of Creative Computing - Volume 2 Feb 1977. (Not in the updated version, published several years latter when the HP-25 was no longer available.) © Creative Computing 1976. Reprinted with permission. Thanks to David Ahl, Editor.

[Curator's note: When James sent this to me I read the first sentence and realized that I had read it two decades ago in The Best of Creative Computing Volume 2 while sitting on my bed with my HP-25 next to me and my Altair 8800 three feet away. I had just flipped the bootstrap code into the Altair and was waiting for it to load Extended BASIC from an audio cassette tape at 30 characters per second - a ten minute process when it worked. Twenty years and 3000 miles of moves later, the Altair and The Best of Creative Computing are still just a few feet away along with several HP-25s.]

Edit Message	Delete Message	Password:

[[Return to the Message Index](#)]



[Go back to the main exhibit hall](#)