50232 Program Description I

Program Title: Direct PIAL compiler

Contributor's Name: P.C.Schmale
Address: Roland Holstlaan 152a
City: Delft Country: Netherlands Postal Code: 2204

Program Description, Equations, Variables: This program accepts a program written in PIAL and converts the PIAL program to numerically coded instructions in the DATA register. These instructions can later be interpreted and executed by the PIAL-EXECUTER. First the structure of PIAL will be explained, so that you can write a program in PIAL.

a) ASSIGNMENT STATEMENTS

You may use 9 variables, designated by the numbers 1 through 9. You may also use the constants 0 through 9, designated by CONST 0 through CONST 9. Available operators (equal priority is executed left to right):
1st priority: ABS, ↑, ARCTAN, SIN, COS, TAN, e↑, LN, Pi, 10↑, INT, CHS, √.
2nd priority: x, / .
3rd priority: +, - .
The form of an assignment statement is algebraic and essentially the same as in ALGOL:
(variable):=(any formula using variables, constants, operators and parentheses not beginning with a left parenthesis);
The symbol := can be read as "becomes". Note the semicolon at the end which may not be omitted. If one or more of the values in the formula must be input during execution, you should substitute the symbol "STOP" on their place. If the result of the formula has to be printed, you should replace (variable) by PRINT. Some examples of legal assignment-statements: 1:=√(CONST 1 - SIN 9↑CONST 2); 2:=STOP + COS (Pi);
PRINT:= e↑4 - e↑(CHS 4);
The STOP statement can also be used outside an assignment statement. The last statement of the program should be STOP.

b) LOOP / JUMP INSTRUCTIONS

A loop can be formed by placing FOR at the beginning and NEXT at the end of a loop. When during execution the program encounters FOR, its location is stored and execution goes on. When NEXT is encountered, execution is taken back to the latest encountered FOR. Escape from the loop is possible with the conditions >xy , =xy , ≠xy and with SKIP. x and y represent a variable (designated by its number). If x and y are the same variable, y is given the value zero. Then is checked if the condition x>y, x=y, x≠y holds. In this case execution goes on. If the condition does not hold, a search is started for the first NEXT. Having found NEXT, the latest FOR is forgotten and the second latest FOR (if there is any) is remembered. Then execution goes on with the statement after NEXT. SKIP has the same effect as a condition that does not hold.

50232 Program Description I

Program Title _____

Contributor's Name _____

Address _____

City _____ Country _____ Postal Code _____

Program Description, Equations, Variables

For instance a loop which must be run through as long as variable 6 > variable 1 would contain the statement > 61 ~~somewhere between FOR and NEXT. With the same set of statements it is~~ possible to create a conditional jump without loop.

IMPLEMENTATION

~~The way in which the calculator accepts the PIAL program is machine-dependent and will be described for the HP97 compiler here:~~

The compiled PIAL program is stored in registers R_{80} through R_{C}, i.e. R_{10} through R_{22}. ~~Before entering a program you must clear the registers~~ and store the begin-adress for the compiled program (mostly 10.0) in R1. Then you add 10^{-5} to R1 (for identification of INT). You press ~~RTN A and you will see the adress in which the next compiled program-~~ step is about to come periodically during a PAUSE statement. During each PAUSE you can enter one symbol. Use for the operators the normal ~~keys, except for ABS. The other symbols are on the card. The condition~~ variables are entered together using MAN (see USER INSTRUCTIONS). While entering you will notice that the adress counts rather irregular. ~~This is because the algebraic form is converted to RPN with the conse-~~ quence that the operator and variable sequence must be changed. The capacity to store pending operations is limited to 5 operators+left ~~parentheses. Also the capacity for pending values is limited to the~~ normal stack capacity of 4. This should give no problems if you don't make the formulas unusual complicated. Some rearranging will solve ~~possible difficulties.~~

A table of the meaning of the codes generated by the compiler is printed under "Sketches". Note that NEXT,+,÷,x,/ only occupy one ~~digit (adress).~~

Operating Limits and Warnings

Don't exceed adress 22.9 (compiled program too large). Trying to enter a function not defined in PIAL results in ~~Error. Other syntactic errors are not detected! After Error you can~~ go on by pressing RTN A. Because the PIAL program is directly compiled, correcting errors is only possible in the compiled program, which is ~~already in RPN form. The MAN entering may be used to subtract wrong~~ codes and add correct. To overcome the error-correcting problem, a PIAL EDITOR and a separate INDIRECT PIAL COMPILER are being develloped.

50232 Program Description II

Sketch(es)

code	meaning	code	meaning	code	meaning
0	NEXT	80	STOP	90	SIN
1	+	81	>	91	COS
2	−	82	=	92	TAN
3	x	83	≠	93	e^x
4	/	84	PRINT	94	LN
5a	STO a	85	FOR	95	Pi
6a	RCL a	86	SKIP	96	10^x
7a	CONST a	87	ABS	97	INT
a stands for 0 through 9.		88	y^x	98	CHS
		89	ARCTAN	99	$\sqrt{x}$

Sample Problem(s) 1) Compile the following program that prints the absolute value of an input number (normally you would use |x| for this purpose)

```
1:= CHS STOP ;
>11
1:= CHS 1;
SKIP
NEXT
PRINT:= CHS 1;
STOP
```

2) Compile the following program that prints SINx/x , with x ranging from 0 to 3.Pi , with steps of Pi/6 .

```
1:= CONST 0;
2:= Pi / CONST 6;
3:= Pi x CONST 3;
```

Solution(s) 1) All the required keystrokes are given here. A comma means that you must wait until the display is steady again (either PAUSE or R/S after pressing A):

CL REG f P≠S * 10.00001 STO 1 RTN A,
1, D, CHS, f d, C, A, B, A, 11 A, 1, D, CHS, 1, C, A, E, f b,
f a, D, CHS, 1, C, f d, R/S.

Now the contents of the program registers should look like this:

R_{10}=8.098518111 ; R_{11}=6.198518606 ; R_{12}=1.988480000 ;
R_{13} through R_{22}=0.000000000 ; * press again f CL REG is recommended.

Reference(s)

50232 Program Description II

Sketch(es)

Sample Problem(s)

```
FOR                                          (example 2 continued)
FOR
=11
PRINT:= CONST 1;
1:= 2;
NEXT
PRINT:= SIN 1 / 1 ;
1:= 1 + 2 ;
>31
NEXT
STOP
```

Solution(s) 2) Required keystrokes: f CL REG f P≷S f CL REG 10.00001 STO 1 RTN A, 1, D, E, 0, C, 2, D, f Pi, /, E, 6, C, 3, D, f Pi, x, E, 3, C, f c, f c, A, C, A, 11 A, f a, D, E, 1, C, 1, D, 2, C, f b, f a, D, SIN, 1, /, 1, C, 1, D, 1, +, 2, C, A, B, A, 31 A, f b, f d, R/S.

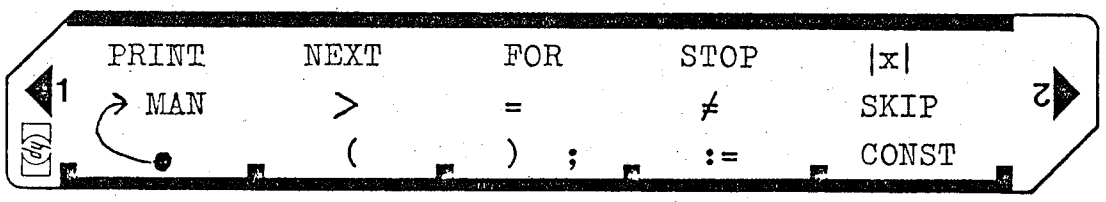
The result in the program register should be:

R_{10}=7.051957645 ; R_{11}=2.957335385 ; R_{12}=8.582117184 ;
R_{13}=6.251061906 ; R_{14}=1.484616215 ; R_{15}=1.813108000 ; others zero.

If you now press f CL REG f P≷S f WRITE DATA, you can record this program on one side of a DATA-card to preserve it for execution with the PIAL EXECUTER.

Reference(s) Also see the remarks to the program PIAL EXECUTER. Specific properties of the executer make program-constructions, not defined here, possible.

50232 User Instructions



STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load side 1 and side 2.			
2	Initialize:			
	For a new PIAL program clear the		f CL REG	
	program registers:		f P⇄S	
	For loading an old piece of PIAL			
	program to which something must be added:			
	enter DATA CRD		f P⇄S	
	Clear working registers:		f CL REG	
	Store begin address in R_1			
	(for a new program add_{begin}=10)	add_{begin}	STO 1	
	Add a small number, e.g. 0.00001 to R_1	10^{-5}	STO +	
	for identification of INT function.		1	
3	Start entering loop		RTN A	
4	During each PAUSE you can enter one of			
	the following symbols:			
	a) CONST		E	
	b) a variable or the value of a const	0 thr.9		
	c) :=		D	
	d) (		B	
	e))		C	
	f) ;		C	
	g) PRINT		f a	
	h) STOP		f d	
	i) an operator:			
	for ABS you use |x|		f e	
	for ↑ you use		y^x	
	for e↑ you use		e^x	
	for 10↑ you use		10^x	
	for $\sqrt{\ }$ you use		$\sqrt{x}$	
	for the other operators you use the			
	corresponding key on the keyboard.			
	j) FOR		f c	
	k) NEXT		f b	
	l) >		A (wait)	
			B	
	m) =		A (wait)	
			C	
	n) ≠		A (wait)	
			D	

1 (first card continued) 2

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
	o) SKIP		A (wait)	
			E	
	p) xy (the condition variables)		A	
		xy	A	
	q) you can also enter one of the			
	compiled instructions under SKETCHES.		A	
		(instr)	A	
	r) subtrackting a wrong compiled code			
	is possible by setting R_1 to its			
	address and entering the negative			
	wrong code (one digit at a time)		A	
		(-)wr.code	A	
	s) if you have entered all of the			
	PIAL program you may press		R/S	
5	If Error occurs during entering, press			
	any key to clear the error and then			
	go to step 3.			
6	To record the compiled program on a			
	DATA-card make the contents of R_0			
	through R_9 zero (If the program does			
	not occupy R_A, R_B, or R_C, you can do			
	this by pressing CL REG).			
	Now press		f W/DATA	
	and you can record the program as			
	data on one side of a card.			

STEP	KEY ENTRY	KEY CODE	COMMENTS
001	001 *LBLA	21 11	beginning of entering loop
	002 *LBL0	21 00	
	003 2	02	
	004 RCL1	36 01	
	005 CF3	16 22 03	
	006 PSE	15 51	entering pause
	007 F3?	16 23 03	check if a key was pressed during pause if not go back for another pause
	008 GTO2	22 02	
	009 STO2	35 02	
010	010 RCL1	36 01	
	011 X=Y?	16-33	
	012 GTO0	22 00	
	013 5	05	
	014 CHS	-22	
	015 STO1	35 46	
	016 GTO5	22 05	
	017 *LBLc	21 16 13	FOR
	018 5	05	
	019 GTO1	22 01	
020	020 *LBLa	21 16 11	PRINT
	021 3	03	
	022 4	04	
	023 STO6	35 06	
	024 GTO0	22 00	
	025 *LBLb	21 16 12	NEXT
	026 CLX	-51	
	027 GSBc	23 16 13	
	028 GTO0	22 00	
	029 *LBLd	21 16 14	STOP
030	030 0	00	
	031 GTO1	22 01	
	032 *LBLe	21 16 15	|x|
	033 8	08	
	034 7	07	
	035 STO5	35 05	
	036 GTO6	22 06	
	037 *LBLB	21 12	enter left parenthesis in operator stack
	038 CLX	-51	
	039 GSBa	23 16 11	
040	040 GTO0	22 00	
	041 *LBLC	21 13	);
	042 *LBL7	21 07	check if operator stack is empty.
	043 RCL0	36 00	
	044 X≠0?	16-42	
	045 GTOC	22 13	
	046 RCL6	36 06	if so, generate a STO instruction if flag 0 has been set.
	047 5	05	
	048 0	00	
	049 +	-55	
050	050 F0?	16 23 00	
	051 GSBc	23 16 13	
	052 CF0	16 22 00	
	053 *LBLC	21 13	empty operator stack into program register until left parenthesis is encountered
	054 GSB3	23 03	
	055 X=0?	16-43	
	056 GTO0	22 00	
	057 GSBc	23 16 13	
	058 GTO7	22 07	
	059 *LBLD	21 14	:= change from STO variables to RCL variables.
	060 SF0	16 21 00	CONST. accept next number as a constant.
	061 GTO0	22 00	
	062 *LBLE	21 15	•→ await another key to be pressed.
	063 SF2	16 21 02	MAN store value in X-register into program register
	064 GTO0	22 00	
	065 *LBLA	21 11	
	066 R/S	51	
	067 *LBLA	21 11	>
	068 GSBc	23 16 13	
	069 GTO0	22 00	
070	070 *LBLB	21 12	
	071 1	01	=
	072 GTO1	22 01	
	073 *LBLC	21 13	
	074 2	02	≠
	075 GTO1	22 01	
	076 *LBLD	21 14	
	077 3	03	
	078 GTO1	22 01	SKIP
	079 *LBLE	21 15	
080	080 6	06	generate a code of the 8a series.
	081 *LBL1	21 01	
	082 8	08	
	083 0	00	
	084 +	-55	
	085 GSBc	23 16 13	
	086 GTO0	22 00	
	087 CLX	-51	generate Error because function cannot be form
	088 ÷	-24	
	089 ∫X	54	
090	090 RTN	24	
	091 CHS	-22	
	092 RTN	24	
	093 INT	16 34	
	094 RTN	24	
	095 10^x	16 33	
	096 RTN	24	
	097 Pi	16-24	try which function key has been pressed
	098 RTN	24	
	099 LN	32	
100	100 RTN	24	
	101 e^x	33	
	102 RTN	24	
	103 TAN	43	
	104 RTN	24	
	105 COS	42	
	106 RTN	24	
	107 SIN	41	
	108 RTN	24	
	109 TAN^-1	16 43	
110	110 RTN	24	
	111 Y^x	31	
	112 RTN	24	

REGISTERS

0	1	2	3	4	5	6	7	8	9
operator-stack	address-counter.	STORE RESULT of KEY PRESSED	in term. ans w in subr. c.	bottom of operator stack in subr. b	latest entered operator.	remember store variable			
S0 PROGR	S1 AM RE	S2 GISTE	S3 RS	S4 PROGR	S5 AM RE	S6 GIST	S7 ERS	S8 PROGRAM	S9

A	B	C	D	E	I
REGISTERS	PROGRAM	REGISTERS			indirect STO RCL GTO

STEP	KEY ENTRY	KEY CODE	COMMENTS
113	÷	-24	(which function key?)
114	RTN	24	
115	X	-35	
116	RTN	24	
117	-	-45	
118	RTN	24	
119	+	-55	
120	RTN	24	
121	*LBL5	21 05	
122	2	02	
123	RCL1	36 01	
124	GSBi	23 45	jump backwards.
125	RCL2	36 02	
126	DSZI	16 25 46	
127	DSZI	16 25 46	
128	X≠Y?	16-32	
129	GTO5	22 05	
130	4	04	
131	RCLI	36 46	
132	CHS	-22	
133	5	05	
134	-	-45	
135	2	02	
136	÷	-24	calculate appropreate function code and store in R5
137	STO5	35 05	
138	X≠Y?	16-35	
139	GTO6	22 06	
140	8	08	
141	3	03	
142	ST+5	35-55 05	
143	*LBL6	21 06	
144	GSB3	23 03	compare priority of latest entered operator P2 with priority of operator that was entered in the operator stack latest P1. (or operator at bottom of stack)
145	STO4	35 04	
146	X=0?	16-43	
147	GTOA	22 11	
148	LN	32	
149	INT	16 34	
150	RCL5	36 05	
151	LN	32	
152	INT	16 34	
153	X>Y?	16-34	
154	GTOA	22 11	
155	RCL4	36 04	If P1 ≥ P2 store operator with P1 in program register, lower stack and compare again.
156	GSBc	23 16 13	
157	GTO6	22 06	
158	*LBLA	21 11	
159	RCL4	36 04	
160	GSBa	23 16 11	If P1 < P2 place operator with P2 at bottom of stack (stack is lifted)
161	RCL5	36 05	
162	GSBa	23 16 11	
163	GTO0	22 00	
164	*LBL3	21 03	lower the operator stack and place the operator that was at the bottom in the X-register
165	EEX	-23	
166	2	02	
167	STX0	35-35 00	
168	RCL0	36 00	

STEP	KEY ENTRY	KEY CODE	COMMENTS
169	INT	16 34	
170	ST-0	35-45 00	
171	RTN	24	
172	*LBLa	21 16 11	place operator in X-register at bottom of stack and lift operator stack.
173	ST+0	35-55 00	
174	EEX	-23	
175	2	02	
176	ST÷0	35-24 00	
177	RTN	24	
178	*LBL2	21 02	number entrance
179	F0?	16 23 00	is it a recall variable?
180	GTOb	22 16 12	
181	STO6	35 06	if not store in R6
182	GTO0	22 00	
183	*LBLb	21 16 12	
184	1	01	
185	0	00	
186	X≠Y	-41	
187	F2?	16 23 02	if it is a constant, add 10 to the code.
188	+	-55	
189	6	06	
190	0	00	form code
191	+	-55	
192	GSBc	23 16 13	
193	GTO0	22 00	
194	*LBLc	21 16 13	store the instruction code in the X-register into the program-register.
195	9	09	
196	X≠Y	-41	
197	X≠Y?	16-35	
198	GTOc	22 16 13	
199	1	01	
200	0	00	
201	÷	-24	
202	STO3	35 03	
203	INT	16 34	
204	GSBc	23 16 13	
205	RCL3	36 03	
206	FRC	16 44	
207	1	01	
208	0	00	
209	X	-35	
210	*LBLc	21 16 13	if the instruction code is 2-number, this part, that stores at one address at a time, is used first as a subroutine, and then as a routine.
211	RCL1	36 01	
212	STOI	35 46	
213	FRC	16 44	
214	1	01	
215	0	00	
216	X	-35	
217	INT	16 34	
218	10^x	16 33	
219	÷	-24	
220	ST+i	35-55 45	
221	.	-62	
222	1	01	
223	ST+1	35-55 01	
224	RTN	24	

LABELS

used 4 x	B ( ) >	C ; =	D : = ≠	E CONST SKIP	0 RCL variables
a PRINT lift op. stack	b NEXT aux.	c FOR Store in pr. reg. aux.	d STOP	e |x|	1
0 begin of entering loop	1 add 80 to code	2 numerical entrance	3 lower operator stack	4	2 entering a constant
5 which key?	6 compare priorities	7 empty operator stack	8	9	3 numerical input

FLAGS

	ON	OFF
0	☐	☒
1	☐	☒
2	☐	☒
3	☐	☒

SET STATUS

TRIG		DISP	
DEG	☒	FIX	☒
GRAD	☐	SCI	☐
RAD	☐	ENG	☐
		n	1