L-9702

MACHINE LANGUAGE INSTRUCTION SET

VERSION TWO

6-21-1972

HENRY K.

UPDATED 10-9-1972

TYPE	MNEM.	CODE 10 9 8 7 6 5 4 3 2 1	DEFINITION	NOTE	ST
1	JSB	← ADDRESS → 0 1	JUMP TO SUBROUTINE GIVEN BY ADDRESS PART		
	BRN	← ADDRESS → 1 1	BRANCH TO ADDRESS ON "FALSE" OR "NO CARRY"		
2	P	← OP. CODE → 0 0 0 1 0	WSP .... POINTER		
	WP	1 0 0	WSWP ... WORD BOUNDARY UP TO POINTER		
	X	0 1 0	WSX ... EXPONENT WITH SIGN		
	XS	1 1 0	WSWX .. EXPONENT SIGN		
	M	0 0 1	WSM ... MANTISSA		
	MS	1 0 1	WSMS ... MANTISSA WITH SIGN		
	W	0 1 1	WSW ... WORD		
	S	1 1 1	WSS ... SIGN OF MANTISSA		
	ZMB	0 0 0 0 0 ← WS →	$\emptyset - B \rightarrow$		
	AMB	1 0 0 0 0	$A - B \rightarrow$		
	SLA	0 1 0 0 0	SHIFT LEFT A-REG		
	AMBA	1 1 0 0 0	$A - B \rightarrow A$ A OR $\overline{B} \rightarrow A$		
	BTC	0 0 1 0 0	$B \rightarrow C$		
	SRB	1 0 1 0 0	SHIFT RIGHT B-REG	CARRY!	
	CTA	0 1 1 0 0	$C \rightarrow A$		
	APBA	1 1 1 0 0	$A + B \rightarrow A$ A OR $B \rightarrow A$		
	AMC	0 0 0 1 0	$A - C \rightarrow$		
	SRC	1 0 0 1 0	SHIFT RIGHT C-REG	CARRY!	
	AMCC	0 1 0 1 0	$A - C \rightarrow C$ A OR $\overline{C} \rightarrow C$	⊛	
	AMCA	1 1 0 1 0	$A - C \rightarrow A$ A OR $\overline{C} \rightarrow A$		
	ZTC	0 0 1 1 0	$\emptyset \rightarrow C$	CARRY!	
	SRA	1 0 1 1 0	SHIFT RIGHT A-REG	CARRY!	
	APCC	0 1 1 1 0	$A + C \rightarrow C$ A OR $C \rightarrow C$	⊛	
	APCA	1 1 1 1 0	$A + C \rightarrow A$ A OR $C \rightarrow A$		
	ZTB	0 0 0 0 1	$\emptyset \rightarrow B$	⊛	
	BXC	1 0 0 0 1	$B \leftrightarrow C$	⊛	
	ATB	0 1 0 0 1	$A \rightarrow B$	CARRY!	
	AXB	1 1 0 0 1	$A \leftrightarrow B$		
	ZMCC	0 0 1 0 1 TEN'S COMPLEMENT	$\emptyset - C \rightarrow C$ $\overline{C} \rightarrow C$	⊛	
	CPCC	1 0 1 0 1	$C + C \rightarrow C$ $C \rightarrow C$	⊛	
	ZMC	0 1 1 0 1 0 9 1 8 2 7 3 6 4 5 5 4 6 3 7 2 8 1 9 0	$\emptyset - C \rightarrow$		
	AXC	1 1 1 0 1	$A \leftrightarrow C$	⊛	
	CM1	0 0 0 1 1	$C - 1 \rightarrow$	⊛	
	AM1	1 0 0 1 1	$A - 1 \rightarrow$		
	CM1C	0 1 0 1 1	$C - 1 \rightarrow C$ C OR $\overline{1} \rightarrow C$ $C \rightarrow C$	⊛	
	AM1A	1 1 0 1 1	$A - 1 \rightarrow A$ A OR $\overline{1} \rightarrow C$ $A \rightarrow C$		
	ZNCC	0 0 1 1 1 NINES COMPLEMENT	$\emptyset - C - 1 \rightarrow C$ $\overline{C}$	⊛	
	ZTA	1 0 1 1 1	$\emptyset \rightarrow A$	CARRY!	
	CP1C	0 1 1 1 1	$C + 1 \rightarrow C$ C OR $1 \rightarrow C$	⊛	
	AP1A	1 1 1 1 1	$A + 1 \rightarrow A$ A OR $1 \rightarrow A$		

0 9 1 8 2 7 3 6 4 5 5 4 6 3 7 2 8 1 9 0

⊛ DAMAGES NON-BCD CHARACTERS IN C-REGISTER

WD	MNEM	CODE 10 9 8 7 6 5 4 3 2 1	DEFINITION	NOTE	IS
3	SS#	←— # —→ 0 0 0 1 0 0	SET STATUS BIT GIVEN BY #		
	RS#	1 0	RESET STATUS BIT GIVEN BY #		
	YS#	0 1	INTERROGATE STATUS BIT #		
	CLS	0 0 0 0 1 1 (NEW CLEAR)	CLEAR ALL STATUS BITS		
	—	x x x x 1 1 (RGS RSD)	15 INSTRUCTIONS NOT AVAILABLE (RGS & RSD)		
4	PT i	←— i —→ 0 0 1 1 0 0	SET POINTER TO VALUE i		
	YP i	1 0	INTERROGATE POINTER AT VALUE i		
		x x x 0 0 1 (NEW POINTER)	8 INSTRUCTIONS; NOT AVAILABLE		
	PRS	x x x 1 0 1	POINTER RIGHT SHIFT (DECREMENT)		
	PLS	x x x 0 1 1	POINTER LEFT SHIFT (INCREMENT)		
		x x x 1 1 1	8 INSTRUCTIONS; NOT AVAILABLE		
5		x x x x 0 0 (ADX POINTER) 1 0 0 0 (EXC POINTER, STATUS BIT OF POINTER SET)	SKIP IF BIF 16 INSTRUCTIONS; (RESERVED)		
	DSTO	0 0 0 0 1 0	DISPLAY TOGGLE		
	DSOF	1 0 0	DISPLAY OFF		
	CTS	0 1 0	STACK UP (C → C → D → E → F; F lost)		
	DNR	1 1 0	CIRCULATE STACK DOWN (C → F → E → D → C)		
	CXM	0 0 1	C ←→ M		
	MTC	1 0 1	M → M; C		
	STA	0 1 1	STACK DOWN (F → F → E → D → A)		
	CLR	1 1 1	CLEAR ALL REGISTERS	BUTB	
		x x x 1 0 0	8 INSTRUCTIONS; NOT AVAILABLE		
	LMD	x x 0 1 1 0 (M1 & M2)	LEARN MODE DISPLAY I_5 → A	LEARN	
		0 0 1 1 1 0		AV.	$\dot{X}_1TC$
		0 1		AV.	X_2TC
		1 0		AV.	X_3TC
		1 1		AV.	X_4TC
	LDC N	←— N —→ 0 1	LOAD CONSTANT N INTO CREG AND PRS		
		x x x 0 1 1 (← MODE LOAD STACK)	8 INSTRUCTIONS; NOT AVAILABLE	STACK	
		x x 0 1 1 1	4 INSTRUCTIONS; NOT AVAILABLE	LMD	
		0 0 1 1 1 1		AV.	X_5TC
	TTC	0 1	T-REG INTO C-REG		
	DSTC	1 0	READ D.S. INTO C-REG		
		1 1		AV.	X_6TC
6	ROM R	←— R —→ 0 0 1 0 0 0 0	ROM R SELECT		
	EERA	x x 0 1 0 (NEW EXT)	EXTERNAL ENTRY TO ROMA		
	TKRA	x x 1 1 (← KEYCODE BUFFER)	TRANSFER KEYCODE TO ROMA		
	RETURN	x x x 0 1 (← RETURN)	RETURN		
	RED	0 0 0 1 1	PREPARE RED RIBBON		A
	ATDS	1 0 0	ADDRESS FROM C TO DATA STORAGE		
	PRE	0 1 0	PRINTER ENABLE		A
	TCS	1 1 0	COMPARE T-REG WITH SECTOR COUNTER		A
	ADV	0 0 1	PAPER ADVANCE		A
	STDS	1 0 1	[illegible]		
	CTT	0 1 1	C → C; T		A
	CCS	1 1 1	COMPARE C-REG WITH SECTOR COUNTER		A

LEVEL	MNEM	CODE 10 9 8 7 6 5 4 3 2 1	DEFINITION	NOTE	SH
7	RMGRA	0 0 0 0 1 0 0 0 0 0	SELECT ROM GROUP A	MAIN D.	1
	RMGRB	0 0 0 1	SELECT ROM GROUP B	F. BLK	1
	RMGRC	0 0 1 0	SELECT ROM GROUP C	D.S.	1
	YADP	0 0 1 1	INTERROGATE AUTO D.P. SWITCH		1
	READ	0 1 0 0	READ PROGRAM MEMORY		1
	SBL	0 1 0 1	SET BUSY LIGHT		1
	YFKB	0 1 1 0	INTERROGATE F. BLK. KBD. FLG, ~~SET BUSY LIGHT AND RESET ERROR LIGHT~~		1
	YPOC	0 1 1 1	INTERROGATE PRINT-ON-COMMAND FLAG		1
		1 0 0 0		AV	
		1 0 0 1		AV	
		1 0 1 0		AV	
		1 0 1 1		AV	
		1 1 0 0		AV	
		1 1 0 1		AV	
		1 1 1 0		AV	
		1 1 1 1		AV	
8	LBR	0 0 0 1 0 0 0 0 0 0	LOAD MEMORY BUFFER INTO ROMA		1
	IS1	0 0 0	ENABLE IS 1		A
	IS2	0 0 1	ENABLE IS 2		A
	IS3	0 1 0	ENABLE IS 3		A
		0 1 1			
		1 0 0			
		1 0 1			
	†INC	1 1 0	INCREMENT PROGRAM COUNTER BY ONE		A
		1 1 1	(RESERVED FOR CHUNG)		
9	MDE	0 0 1 0 0 0 0 0 0 0	DELETE CURRENT MEMORY CELL	LEARN	1
	MIN	0 1	INSERT BUFFER INTO MEMORY	LEARN	1
	CLM	1 0	CLEAR MEMORY	LEARN	1
	SRH	1 1	SEARCH FOR LABEL	LEARN	1
10		0 1 0 0 0 0 0 0 0 0	(RESERVED FOR CHUNG)		
	PAD	1	POINTER ADVANCE BY ONE	LEARN	1
11	TTP	1 0 0 0 0 0 0 0 0 0	LOAD T-REG INTO PROGRAM COUNTER		A
12	DUMMY	1 0 0 0 0 0 0 0 0 0			A

TYPE	MNEM	CODE 10 9 8 7 6 5 4 3 2 1	DEFINITION	NOTE	TS
21	LIOn	← n → 1	LOAD n (9-BITS) INTO I/O REGISTER		2
22		x x x x x x x x 1 0	NOT AVAILABLE (W.S. TROUBLE)		2
23		x x x x x x x 1 0 0	(RESERVED)	dv.	2
24		0 0 0 0 0 0 1 0 0 0			
		1 0			
		0 1			
	ADD	1 1	BINARY ADD T ⊕ C → T		
		1 0 0 0 0			
	PDEC	1 0	DECREMENT PRGM. COUNTER BY ONE		2
	PTT	0 1	LOAD PRGM. COUNTER INTO T-REG		2
		1 1			
	SLT	0 1 0 0 0	SHIFT T-REG LEFT ONE BIT		2
	SRT	1 0	SHIFT T-REG RIGHT ONE BIT		2
	IXT	0 1	EXCHANGE I/O REG DATA WITH T-REG		2
	SRI	1 1	9-BIT SHIFT RIGHT IN I/O-REG		2
	TINC	1 1 0 0 0	INCREMENT T-REG BY ONE (BIN)		2
	TDEC	1 0	DECREMENT T-REG BY ONE (BIN)		
		0 1			
		1 1			
	XOR	0 0 1 0 0	EXCLUSIVE OR T ∀ C → T		2
	IOR	1 0	INCLUSIVE OR T ∨ C → T		2
	AND	0 1	LOGICAL AND T ∧ C → T		2
		1 1			
		1 0 1 0 0			
		1 0			
		0 1			
		1 1			
		0 1 1 0 0			
		1 0			
		0 1			
		1 1			
	YBC	1 1 1 0 0	INTERROGATE AND RESET BIN. CARRY	TWICE	2
		1 0			
		0 1			
		1 1			
		x x x x x 1	(RESERVED)		
25		x x x 0 0 1 0 0 0 0	NOT AVAILABLE (ROM SELECT TRBLE)		
		x x x 1 0	NOT AVAILABLE		
		x x x 0 1	NOT AVAILABLE		
	RED	0 0 0 1 1	PREPARE RED RIBBON		4
		1 0 0			
	PRE	0 1 0	PRINTER ENABLE		A
	TCS	1 1 0	COMPARE T-REG WITH SECTOR COUNT		A
	ADV	0 0 1	PAPER ADVANCE		A
		1 0 1			
	CTT	0 1 1	C → C, T		A
	CCS	1 1 1	COMPARE C-REG WITH SECTOR COUNTER		A

OCT	MNEM	CODE 10 9 8 7 6 5 4 3 2 1	DEFINITION	NOTE	SH
26	YINTE	0 0 0 0 1 0 0 0 0 0	INTERROGATE FLAG		2
	WRITE	0 0 1	WRITE PROGRAM INSTRUCTION		2
	RSL	0 1 0	LEFT BUSY LIGHT		2
	GIOE	0 1 1	GENERAL I/O - DEVICE ENABLE		2
		1 0 0			2
	~~ELOE~~	~~1 0 1~~	~~ERROR LIGHT ON~~		2
	TG9	1 1 0	GENERAL I/O INSTRUCTION ONE		2
	TG8	1 1 1	GENERAL I/O INSTRUCTION TWO		2
		x x x 1	RESERVED		2
27		0 0 0 1 0 0 0 0 0 0 0	NOT AVAILABLE		2
	IS1	1 0 0	ENABLE IS1		A
	IS2	0 1 0	ENABLE IS2		A
	IS3	1 1 0	ENABLE IS3		A
		0 0 1			
		1 0 1			
	PINC	0 1 1	INCREMENT PROGRAM COUNTER BY ONE		A
		1 1 1			
28		0 0 1 0 0 0 0 0 0 0			
		1 0			
		0 1			
		1 1			
29		0 1 0 0 0 0 0 0 0 0			
		1			
30	TTP	1 0 0 0 0 0 0 0 0 0	LOAD T-REG INTO PROGRAM COUNTER		A
31	DUMMY	0 0 0 0 0 0 0 0 0 0			A

L-9702

6-20-1972.

EXECUTION LANGUAGE INSTRUCTION CODES AND OTHER USEFUL STARTING ADDRESSES.

MAIN UNIT SOFTWARE VERSION 2.

NOTES :
- INSTRUCTION CODE (8-BITS WIDE) DEFINES EXECUTION ROUTINE STARTING ADDRESS ON ROM A-Ø
- K ... FUNCTION ASSOCIATED WITH MAIN UNIT KEYBOARD
- SK.. FUNCTION ASSOCIATED WITH SHIFTED MAIN UNIT KEYBOARD
- S .. SINGLE OPERAND FUNCTION
- T .. TWO OPERANDS FUNCTION
- A .. ARITHMETIC STATEMENT CONTROL FUNCTION
- P .. PROGRAM CONTROL FUNCTION
- F .. SPECIAL FUNCTION
- D .. DATA ENTRY FUNCTION

HENRY K.

000		PWON	POWER-ON	F
064		START	START PROGRAM EXECUTION	
012	÷	DIV	FLOAT. POINT DIVISION	K T
013	*	MUL	FLOAT. POINT MULTIPLICATION	K T
014	↑	XTY	POWER FUNCTION	K T
016	)	RPAR	RIGHT PARENTHESIS	K A
020	=	EQUAL	EQUAL	K T
022	.	DECP	DECIMAL POINT	K D
023	0	ZERO	DIGIT ZERO	K D
026	(	LPAR	LEFT PARENTHESIS	K A
032	3	THREE	DIGIT "3"	K D
033	2	TWO	DIGIT "2"	K D
034	1	ONE	DIGIT "1"	K D
036	%	PERC	PERCENT	K S
040	+	ADD	FLOAT. POINT ADDITION	K I
042	6	SIX	DIGIT "6"	K D
043	5	FIVE	DIGIT "5"	K D
044	4	FOUR	DIGIT "4"	K D
046	←	RCLEX	RESTORE DATA FROM DEDICATED REGISTER	K S
050	−	SUB	FLOAT. POINT SUBTRACTION	K T
052	9	NINE	DIGIT "9"	K D
053	8	EIGHT	DIGIT "8"	K D
054	7	SEVEN	DIGIT "7"	K D
056	→	STOREX	STORE DATA IN DEDICATED REGISTER	K S
062		RUN	CONTINUE IN PROGRAM EXECUTION	K P
063		LENTRY	SET LAST ENTRY FLAG.	K P
066		CANCEL	CANCEL ENTRY	K D
070		SHIFT	SHIFT	K F
072		RND( )	ROUND	K F
073		RESET	RESET	K F
076		PRINTEX	PRINT DATA	K S

150		JMPRB	JUMP RELATIVE BACKWARD	P
107		NOP	PROGRAMMED NOP	P
131		CHS	CHANGE SIGN OF DATA	SK D
207		LABEL	LABEL HEAD BEGINNING	P
152		JMPRF	JUMP RELATIVE FORWARD	P
224		JSBLC	JSB TO COMPUTED LABEL	P
225		JSBLP	JSB TO LABEL	P
226		JSBABS	JSB TO ADDRESS	P
240			F.BLOCK FUNCTION # 1	K
241			# 2	K
242			# 3	K
243			# 4	K
244			# 5	K
245			# 6	K
246			# 7	K
247			# 8	K
250			# 9	K
251			# 10	K
252			# 11	K
253			# 12	K
254			# 13	K
255			# 14	K
256			# 15	K
263		STOP	STOP PROGRAM EXECUTION	P
267	$1/x$	ONEX	ONE OVER X	SK S
270	e^x	EXP	EXPONENTIAL FUNCTION	SK S
275		IFZERO	IF DATA IS ZERO	P
300		STEP	PROGRAM EXECUTION STEP	P
312		IFPOS	IF DATA IS POSITIVE	P
316		IFNEG	IF DATA IS NEGATIVE	P
322		IFLE	IF LAST ENTRY FLAG SET	P
236		GRTOT	GRAND TOTAL	SK S
237		TABLE	SPECIAL F. BLOCK FUNCTION	SK F
134	T3	PAGE	GO TO NEXT PRGM. MEMORY PAGE	P

325	T1	LIST	LIST DATA STORAGE CONTENTS	T
326	T∅	CLRDATA	CLEAR DATA STORAGE REGISTERS	T
327		DSINC	INCREMENT D.S. GIVEN BY <C>	T
330		DSDEC	DECREMENT D.S. GIVEN BY <C>	T
331		STODIR	STORE DIRECT IN D.S. ADDR. ( )	T
332		ACCPLUS	CHANGE STORE TO ACC +	T
333		ACCSUB	CHANGE STORE TO ACC −	T
334		ACCMUL	CHANGE STORE TO ACC ×	T
335		ACCDIV	CHANGE STORE TO ACC ÷	T
336		RECALL	RECALL FROM D.S. ADDRESS ( )	T
337		EXCH	EXCHANGE WITH DATA AT ADDR ( )	T
342		RETURN	RETURN FROM SUBROUTINE	P
346		SETFLG	SET FLAG ( )	P
347		IFFLG	IF FLAG ( ) SET	P
353		GTCOMPL	GO TO COMPUTED LABEL	P
354		GTLABEL	GO TO LABEL	P
355		GTABS	GO TO ADDRESS	P
362	×/12	XOTW	X OVER TWELVE	SK S
363	T4	PAPADV	PAPER ADVANCE	F
373		END	PROGRAM END	P
374	LOG	LOGEX	COMMON LOG.	SK S
375	LN	LNEX	NATURAL LOG.	SK S
376		IOCALL	I/O CHANNEL ( ) CALL	SK F
340		DSTEST	DECREMENT AND TEST IF ZERO	P

6-27-72

CONDITIONAL INSTRUCTIONS

(A) LIST OF CONDITIONAL INSTRUCTIONS:

JMPRF	152	JUMP RELATIVE FORWARDS	(SKIP F.)
JMPRB	150	JUMP RELATIVE BACKWARDS	(SKIP B.)
JSBLC	224	JSB TO COMPUTED LABEL	
JSBLP	225	JSB TO LABEL	
JSBABS	226	JSB TO ADDRESS	
GTCOMPL	353	GO TO COMPUTED LABEL	
GTLABEL	354	GO TO LABEL	
GTABS	355	GO TO ADDRESS	
RETURN	342	RETURN FROM SUBROUTINE	

(B) LIST OF CONDITION GENERATING TESTS:

IFZERO	275	IF DATA IS ZERO
IFPOS	312	IF DATA IS POSITIVE
IFNEG	316	IF DATA IS NEGATIVE
IFLE	322	IF LAST ENTRY FLAG SET
IFFLG #	347	IF FLAG # SET (#: 0; 1; 2; ---; 9)

(C) NOTES:

- LISTED CONDITIONAL INSTRUCTIONS ARE EXECUTED IF CONDITIONS IS MET; OTHERWISE THE INSTRUCTION IS SKIPED
- THERE IS NO NEED TO PROGRAM CONDITIONAL INSTRUCTION IMMEDIATELY AFTER TEST. THE RESULT OF THE TEST IS INTERNALY STORED AND APPLIED TO FIRST CONDITIONAL INSTRUCTION FOUND.
- CONDITIONAL INSTRUCTION NOT PRECEDED BY TEST IS INTERPRETED AS UNCONDITIONAL ONE AND DIRECTLY EXECUTED.
- TEST FOR CONDITIONS GIVEN BY PROGRAM FLAGS OR LAST ENTRY FLAG ALSO RESET TESTED FLAG TO ZERO.

FORMATED INSTRUCTIONS IN THE EXECUTION LANGUAGE

① PRINT ALPHA MESSAGE FROM THE PROGRAM MEMORY

Program address		
N	IOCALL	$376_{(8)}$
N+1	ZERO	$023_{(8)}$
N+2	LPAR	$026_{(8)}$
N+3	C_{18} C_{17}	M_1
N+4	C_{16} C_{15}	M_2
N+5	C_{14} C_{13}	M_3
N+6	C_{12} C_{11}	M_4
N+7	C_{10} C_9	M_5
N+8	C_8 C_7	M_6
N+9	C_6 C_5	M_7
N+10	C_4 C_3	M_8
N+11	C_2 C_1	M_9
N+12	RPAR	$016_{(8)}$
N+13	LENTRY	$063_{(8)}$
new instruction		

WHERE M_1 THROUGH M_9 ARE 8-BIT CODES REPRESENTING THE MESSAGE. EACH OF THESE 8-BIT CODES CONSISTS OUT OF TWO FOUR-BIT CODES REPRESENTING SECTOR NUMBER FOR GIVEN PRINT DRUM DESIGN.

E.G. TO PRINT MESSAGE:

COLUMNS: 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

MESSAGE: E N T E R ␣ D A T A ␣ A =

THE SEQUENCE OF PROGRAM INSTRUCTIONS WILL BE:

* E	N+3	$C_{18}=13$	$C_{17}=11$	⇒ 11011011	$M_1=333_{(8)}$
N T	N+4	$C_{16}=12$	$C_{15}=10$	⇒ 11001010	$M_2=312_{(8)}$
E R	N+5	$C_{14}=11$	$C_{13}=11$	⇒ 10111011	$M_3=273_{(8)}$
* *	N+6	$C_{12}=13$	$C_{11}=13$	⇒ 11011101	$M_4=335_{(8)}$
D A	N+7	$C_{10}=11$	$C_9=11$	⇒ 10111011	$M_5=273_{(8)}$
T A	N+8	$C_8=11$	$C_7=11$	⇒ 10111011	$M_6=273_{(8)}$
* *	N+9	$C_6=13$	$C_5=13$	⇒ 11011101	$M_7=335_{(8)}$
* A	N+10	$C_4=13$	$C_3=12$	⇒ 11011100	$M_8=334_{(8)}$
* =	N+11	$C_2=13$	$C_1=2$	⇒ 11010010	$M_9=322_{(8)}$

WHERE "13" IS USED TO CODE "BLANK" FOR GIVEN COLUMN. *

NOTE: - addresses N; N+1; N+3 THROUGH N+11 contain information for execution
- addresses N+2; N+12; N+13 contain syntax terminators to avoid problems IN LIST-mode.

(2) LABEL NUMBER FORMAT

IS USED AS ADDITIONAL INFORMATION NEEDED FOR EXECUTION OF JSBLP, GTLABEL, JSBLC INSTRUCTIONS

FORMATS:

a – LABEL HEAD	N	LABEL	151 (8)
	N+1	L_1 L_2	M_1
	N+2	L_3 ∅	M_2
b – JSBLP	N	JSBLP	225 (8)
	N+1	L_1 L_2	M_1
	N+2	L_3 ∅	M_2
c – JSBLC	N	JSBLC	224 (8)
d – GTLABEL	N	GTLABEL	354 (8)
	N+1	L_1 L_2	M_1
	N+2	L_3 ∅	M_2

WHERE: L_1; L_2; L_3 ARE BCD-CODED DECIMAL DIGITS.
L_1 --- MOST SIGNIFICANT DIGIT OF LABEL)
L_3 -- LEAST SIGNIFICANT DIGIT OF LABEL)
AND ∅ STANDS FOR FOUR ZERO BITS
L_1; L_2; L_3 = ∅, 1, 2, ---, 9.

EXAMPLE: JUMP TO SUBROUTINE WITH LABEL 702:

N	JSBLP		225 (8)
N+1	7 0	01110000	M_1 = 160 (8)
N+2	2 ∅	00100000	M_2 = 040 (8)

where $L_1 = 7$
$L_2 = 0$
$L_3 = 2$

NOTE: GTCOMPL IS ONE 8-BIT INSTRUCTION ONLY!

6-26-72

(3) ABSOLUTE ADDRESS FORMAT

IS USED TO INDIGATE POINT IN THE PROGRAM WHERE THE PROGRAM EXECUTION WILL CONTINUE WITH NEXT INSTRMCTION. THE ADDRESS IS ALWAYS GIVEN BY 10-BIT BINARY NUMBER.

FORMATS:

a - GTABS

	GTABS	
N		355 (8)
N+1	$\emptyset\emptyset\ B_9\ B_8\ B_7\ B_6\ B_5\ B_4$	M_1
N+2	$B_3\ B_2\ B_1\ B_0\ \emptyset\emptyset\emptyset\emptyset$	M_2

6 - JSBABS

	JSBABS	
N		226 (8)
N+1	$\emptyset\emptyset\ B_9\ B_8\ B_7\ B_6\ B_5\ B_4$	M1
N+2	$B_3\ B_2\ B_1\ B_0\ \emptyset\emptyset\emptyset\emptyset$	M_2

WHERE $B_0\ B_1$ -- B_9 ARE BITS OF THE ADDRESS;
B_0 --- LEAST SIGNIFICANT BIT,
B_9 --- MOST SIGNIFICANT BIT,
AND $\emptyset$ STANDS FOR BITS WHICH MUST BE ZERO.

NOTE: DO NOT FORGET THAT FIRST INSTRMCTION TO BE EXECUTED AFTER THESE JUMPS IS LOCATED AT ADDRESS GIVEN BY $M_1\ M_2$ PLUS ONE.

EXAMPLE: NEXT INSTRMCTION TO BE EXECUTED IS LOCATED AT ADDRESS A= 256; GO TO THAT ADDRESS.

$A = 256$
$A-1 = 255 \Rightarrow$ $B_9 = \emptyset$ $B_8 = \emptyset$ $B_7 = 1$
$B_6 = 1$ $B_5 = 1$ $B_4 = 1$
$B_3 = 1$ $B_2 = 1$ $B_1 = 1$
$B_0 = 1$

	GTABS		
N			355 (8)
N+1	$\emptyset\emptyset\emptyset\emptyset\ 1\ 1\ 1\ 1$	$M_1 =$	017 (8)
N+2	$1\ 1\ 1\ 1\ \emptyset\emptyset\emptyset\emptyset$	$M_2 =$	360 (8)

NOTE: GTABS & JSBABS FOR TARGET ADDRESS ZERO ARE NOT ALLOWED

6-26-72

(4) RELATIVE JUMPS (SKIPS)

ALLOW TO SKIP BACKWARDS OR FORWARDS DESIRED NUMBER OF PROGRAM STEP. THESE ROUTINES ARE EXECUTED BY ADDING THE BINARY NUMBER REPRESENTING TO SIZE OF THE JUMP TO CURRENT PROGRAM COUNTER CONTENTS. IT MEAN THAT SKIP FORWARDS ARE ACCOMPLISHED BY ADDING A POSITIVE BINARY NUMBER; SKIP BACKWARDS BY ADDING OF A COMPLEMENTED NUMBER.

NOTE: DO NOT FORGET THAT FIRST INSTRUCTION TO BE EXECUTED AFTER THESE SKIPS IS LOCATED AT COMPUTED ADDRESS PLUS ONE

FORMATS:

NUMBER OF INSTRUCTIONS ACTUALLY SKIPPED IS COUNTED FROM LOCATION OF JMP HERE

a- SKIP FORWARD

	JMPRF	
N		152 (8)
N+1	$\emptyset\emptyset B_9 B_8 B_7 B_6 B_5 B_4$	M_1
N+2	$B_3 B_2 B_1 B_0 \emptyset \emptyset \emptyset \emptyset$	M_2

a- SKIP BACKWARD

	JMPRB	
N		150 (8)
N+1	$\emptyset\emptyset B_9 B_8 B_7 B_6 B_5 B_4$	M_1
N+2	$B_3 B_2 B_1 B_0 \emptyset \emptyset \emptyset \emptyset$	M_2

WHERE $B_9, B_8, \ldots, B_0$ ARE BITS OF THE BINARY ADDRESS MODIFICATOR;
B_0 --- LEAST SIGNIFICANT DIGIT,
B_9 -- MOST SIGNIFICANT DIGIT,
AND $\emptyset$ STAND FOR BITS WHICH MUST BE ZERO.

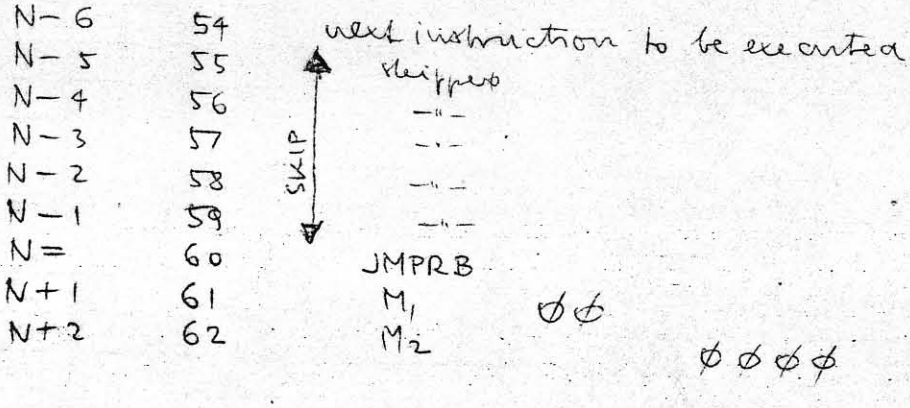
A- EXAMPLE: SKIP NEXT FIVE INSTRUCTIONS.

N =	55	JMPRF		152 (8)
N+1 =	56	M_1	$\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset$	M_1 = 000 (8)
N+2 =	57	M_2	$\emptyset 1 \emptyset 1 \emptyset\emptyset\emptyset\emptyset$	M_2 = 120 (8)
N+3 =	58	skipped with		
N+4 =	59			
N+5 =	60			
N+6 =	61	next instruction to be executed		

SKIP

current address:	55 :	0110111
modifier	+ 5 :	0000101
result address	60 :	0111100
code to be exec. is on:	61 :	0111101

B – EXAMPLE : SKIP FIVE INSTRUCTION BACKWARDS



current address	60 :	0 111 100	
modifier		1 111 001	(≡ SKIP + 1 complemented)
result address to be executed	53 :	0 110 101	
	54 :	0 110 110	

(5) PROGRAM FLAGS.

THERE IS TEN PROGRAM FLAGS (Ø, 1, -- , 9) TO ALLOW CONDITIONAL FLOWS IN PROGRAMS.

FORMATS:

9 – SET FLAG

```
SETFLG          346 (8)
NUMBER          M1
```

6 – INTERROGATE AND RESET FLAG

```
IFFLG           347 (8)
NUMBER          M1
```

WHERE "NUMBER" STANDS FOR ANY DEC. DIGIT.

EXAMPLE: TEST FLAG "6"

```
N       IFFLG       347 (8)
N+1     SIX         042
```

8-30-1972

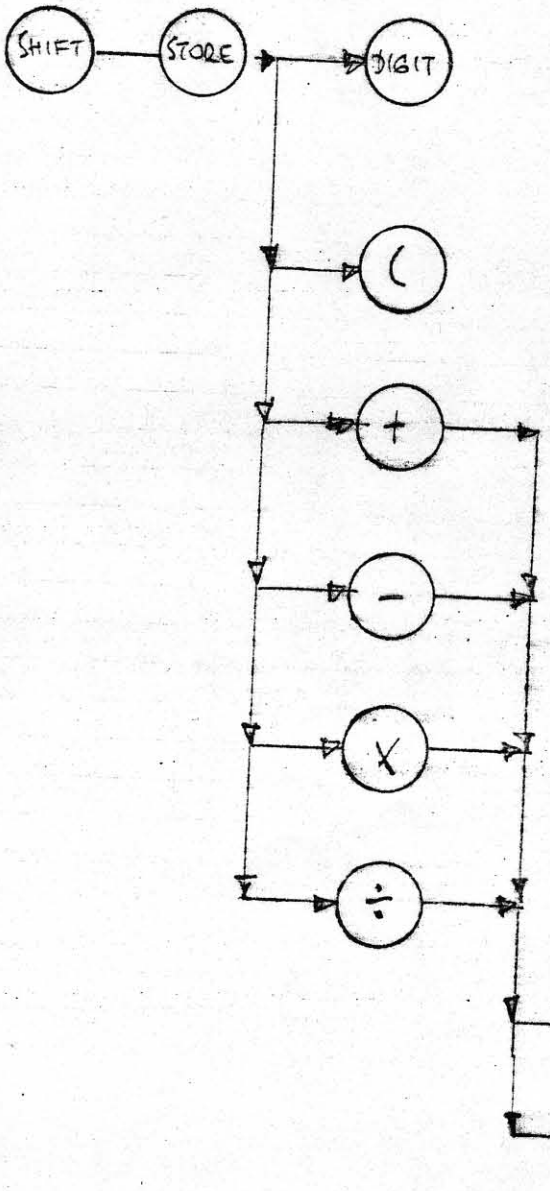
DATA STORAGE GROUP SOFTWARE

CONTROL PART OF THE DATA STORAGE SOFTWARE IS CURRENTLY LOCATED IN ROM GROUP C, ROM: Ø, 1, 2. THE EXECUTION PART OF DATA STORAGE ROUTINES ARE PART OF THE MAIN UNIT SOFTWARE.

FUNCTIONS

(A) STORE IN D.S. REGISTER

GENERALLY THIS IS A TWO OPERAND (DIADIC) OPERATOR WHERE D.S. ADDRESS SERVES FOR THE SECOND OPERAND. IT IS INITIATED BY KEY SEQUENCE SHIFT STORE. IF FOLLOWED BY DIGIT OR LEFT PARENTHESIS IT IS INTERPRETED AS STORE DIRECT, WHEN FOLLOWED BY OPERATOR (+ − × ÷) IS INTERPRETED AS ACCUMULATION:



STORE DIRECT IN D.S. ADDRESS GIVEN BY NUMBER BEING ENTERED INTO GREG

⟨D⟩ → D.S. ; C ; D

STORE DIRECT IN D.S. ADDRESS GIVEN BY ARITHMETIC EXPRESSION IN PARENTHESES

⟨DS⟩ + ⟨D⟩ → D.S
⟨D⟩ → C;D
} ACCUMULATE +

⟨DS⟩ − ⟨D⟩ → D.S.
⟨D⟩ → C;D
} ACCUMULATE −

⟨DS⟩ × ⟨D⟩ → D.S.
⟨D⟩ → C;D

⟨DS⟩ ÷ ⟨D⟩ → D.S.
⟨D⟩ → C;D

DIGIT FOR ADDRESS GIVEN BY NUMBER

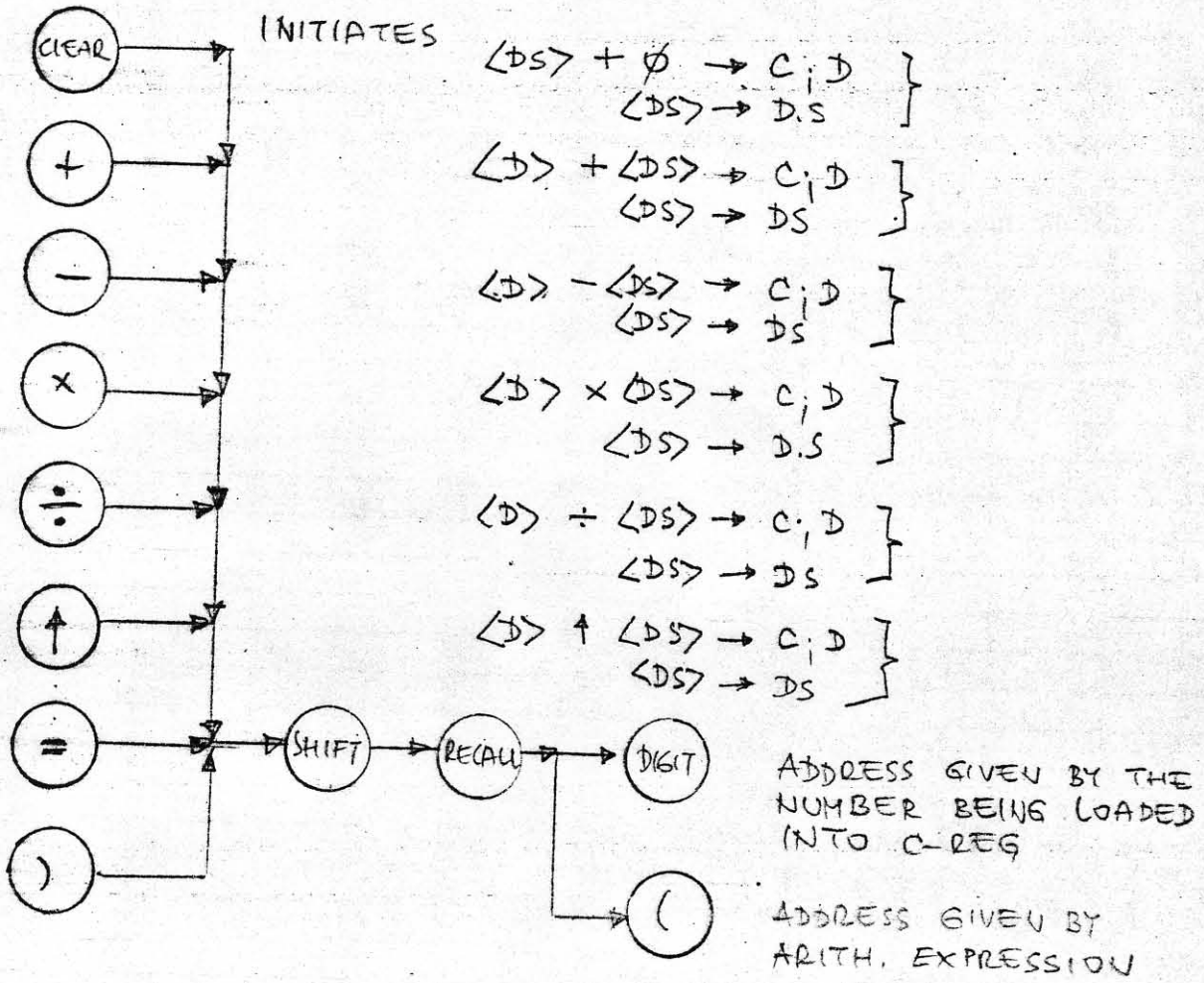
(FOR ADDRESS GIVEN BY ARITH. EXPRESSION

OTHER KEYS WILL BE REJECTED BY THE MACHINE (IN THIS SEQUENCE)

IF CALLED D.S. ADDRESS IS ILLEGAL FOR GIVEN SYSTEM, ERROR MESSAGE "NOTE 26" IS PRINTED TO INDICATE THAT CALLED REGISTER WAS NOT FOUND.

(B) RECALL DATA FROM D.S.

TAKES IN ACCOUNT PREVIOUSLY KEYED OPERATION. IT IS AGAIN A DIADIC OPERATOR



NOTE: RECALL AFTER (=) OR ( ) IS PLAIN RECALL DIRECT.

USING PARENTHESES RECALL INDIRECT IS POSSIBLE. e.g.

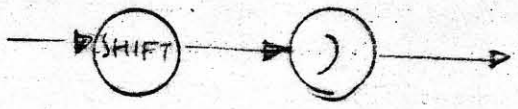
(SH) (RCL) ((SH) (RCL) (1) (9) ( ) (=)

WILL RECALL NUMBER FROM REGISTER GIVEN BY THE NUMBER STORED IN D.S. 19.

(C) EXCHANGE

IS AGAIN GENERALLY DIADIC OPERATOR WITH ADDRESS AS SECOND OPERAND.
ADDRESS AGAIN CAN BE DEFINED DIRECTLY BY THE FOLLOWING NUMBER OR BY THE FOLLOWING ARITHMETIC EXPRESSION

IT IS INITIATED BY KEY SEQUENCE:

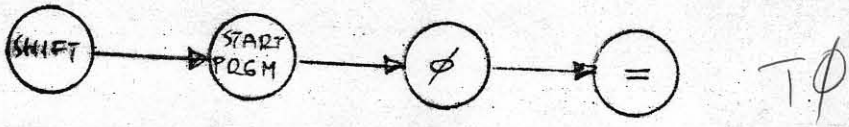


AND PERFORMS OPERATION:

$\langle C \rangle \rightarrow DS$.
$\langle DS \rangle \rightarrow C$

(D) CLEAR DATA

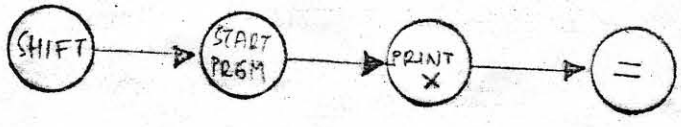
IF COMPLETE RESET OF THE WHOLE D.S. CAPACITY IS REQUIRED, THE FOLLOWING KEY SEQUENCE WILL INITIATE PROPER ROUTINE:



THIS ROUTINE DOES NOT CHANGE SYNTAX SITUATION NOR DATA IN CPU.

(E) LIST DATA

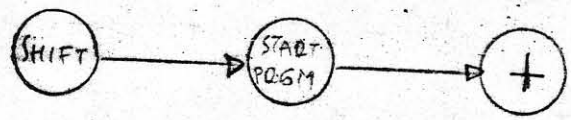
ROUTINE PRINTS CONTENTS OF ALL D.S (STARTING FROM ADDRESS Ø) IN GROUPS OF TEN NUMBERS. ADDRESSES ARE AUTOMATICALLY INCREMENTED BY ONE. THE FIRST ADDRESS NOT FOUND CAUSES END OF THE ROUTINE. INITIATION SEQUENCE:



EXECUTION OF THIS ROUTINE CAN BE ARTIFICIALLY BROUGHT TO THE END BY DEPRESSING OF THE "STOP" KEY

(F) INCREMENT, DECREMENT

ANY DATA STORAGE REGISTER CONTENTS (ADDRESS GIVEN BY C-REGISTER CONTENTS) CAN BE INCREMENTED BY ONE OR DECREMENTED BY ONE WITHOUT CHANGING CURRENT SYNTAX OR CPU DATA. PROPER ROUTINES ARE INITIATED BY SEQUENCES:



FOR INCREMENT, OR

FOR DECREMENT.

(G) INSTRUCTION CODES ASSOCIATED WITH DATA STORAGE SOFTWARE:

OCTAL:

325	LIST DATA
326	CLEAR DATA
327	D.S. INCREMENT
330	D.S. DECREMENT
331	STORE IN D.S.
332	MODIFIES "STORE" INTO ACC + (+)
333	MODIFIES "STORE" INTO ACC − (−)
334	MODIFIES "STORE" INTO ACC × (×)
335	MODIFIES "STORE" INTO ACC ÷ (÷)
336	RECALL FROM D.S.
337	EXCHANGE WITH D.S.

e.g. TO ACCUMULATE X CURRENT RESULT WITH CONTENTS OF REGISTER 15:

```
  :
331
334
034
043
020
  |
  |
  )
```

STORE
(×)
1
5
=

(causes actual execution)

(H) ADDRESS

The hardware understands by "address" the integer part of the absolute value of a dat (in normalized floating point notation) residing in C-register.
The software does not change the actual contents of the C-register and prints it in the same way as other data

e.g.:

STORE 9999 → ∅
STORE 8888 → 1.50
STORE 7777 → 2.99
STORE 6666 → −3.01

```
                CLEAR
DATA              =

 .00
 .00
 .00
 .00
 .00
 .00
 .00
 .00
 .00
 .00

 .00
 .00
 .00
 .00
 .00
 .00
```

```
                CLEAR
9999.00          S→
    .00           =
9999.00           T
8888.00          S→
   1.50           =
8888.00           T
7777.00          S→
   2.99           =
7777.00           T
6666.00          S→
   3.01           =
6666.00           T
DATA              =

9999.00
8888.00
7777.00
6666.00
 .00
 .00
 .00
 .00
 .00
 .00
```

DSTEST

(I) DECREMENT AND TEST IF ZERO

This function decrements D.S. register (address given by the number in C-reg) and tests if result is zero.
If ZERO then flag is set to indicate condition met. Else flag is set to indicate condition not met
Decremented number is stored back in D.S.

This function can be used only in program run mode. It is not accessible in manual control mode

Execution routine of this function does not change previous result in D-reg

Starting address 340 (OCTAL)

11-6-1972

9805 MAIN UNIT - PRINTED MESSAGES

CLEAR	INDICATES READINESS OF THE MACHINE AFTER PWON, RESET, START PROGRAM	
DATACLEAR	INDICATES COMPLETITION OF THE DATA STORAGE RESET ROUTINE	
DATA =	HEAD OF THE DATA STORAGE REGISTERS CONTENTS LIST	
END	END OF THE DATA STORAGE REGISTERS CONTENTS LIST	
NOTE ØØ	TO MANY SUBROUTINE CALLS OR RETURNS	RECOVERY: RESET
NOTE Ø4	FIVE PARENTHESIS LEVELS ALREADY ASSIGNED	LOAD NUMBER etc.
NOTE Ø5	RIGHT PARENT. NOT PRECEEDED BY A LEFT ONE	OPERATOR, =, LEFT PARENTH.
NOTE Ø8	NEG. MANTISSA TO NOT INTEGER POWER ×	RELOAD INTEGER EXPONENT
NOTE Ø9	NEG X IN LOG OR LN ROUTINES ×	RELOAD X
NOTE 11	a - BEFOR RESULT MEANS REGULAR OVERFLOW	OPERATOR etc.
	b - AFTER RESULT MEANS GT-REG. OVERFLOW	OPERATOR etc.
NOTE 12	X≡Ø IN LOG AND LN ROUTINES	RELOAD X
NOTE 15	X≡Ø TO NEGATIVE POWER	OPERATOR etc.
NOTE 16	ZERO DIVISOR	OPERATOR etc.
NOTE 2Ø	LABEL NOT FOUND	RESET
NOTE 21	a - I/O DEVICE NOT FOUND	CALL PROPER DEVICE
	b - END OF THE PROGRAM MEMORY REACHED	RESET
	c - PROGRAM MEMORY NOT READY	RESET
	d - DS SOFTWARE NOT READY (MANUAL MODE)	OPERATOR, LEFT PARENTH, etc.
	e - FUNCTION BLOCK NOT READY (RUN MODE)	RESET
	f - DS SOFTWARE NOT READY (RUN MODE)	RESET
NOTE 22	RIGHT PARENTESIS AFTER AN OPERATION	LOAD DATA, etc.
NOTE 24	EQUAL INSIDE PARENTHESES	OPERATOR, =, LEFT. PARENTH
NOTE 26	D.S. ADDRESS NOT FOUND	2× CANCEL ENTRY, RELOAD ADDRESS

COMMENT:
LISTED RECOVERY PROCEDURES ARE ONLY BASIC ONES. WITH KNOWLEDGE OF THE EXECUTION LANGUAGE SYNTAX AND BY TAKING IN ACCOUNT DISPLAYED DATA RECOVERY PROCEDURE CAN BE FOUND TO FIT MANY SITUATION. RECOVERY FROM SYNTAX ERRORS IN RUN MODE HAS MEANING ONLY VIA PROGRAM CHANGES.

MESSAGES ARE PRINTED IN BOTH COLORS

MAIN UNIT INTERNAL DATA STORAGE REGISTERS

ADDRESS FORMAT:

PT: 0 0 IS n 0 0 0 0 0 0 0 0 1

ADDR: 13 12 11 10 9 8 7 6 5 4 3 2 1 0

0 15 n 0 0 0 0 0 0 0 0 0 0 1

Where n = 0; 1; 2; ... ; 9

REGISTERS:

n = 0	OP. CODE STACK FOR PARENTHESIS OPERATIONS (3)
n = 1	DATA STACK FOR 1st LEVEL IN PAR. OPER.
n = 2	DATA STACK FOR 2nd LEVEL IN PAR. OPER.
n = 3	DATA STACK FOR 3rd LEVEL IN PAR. OPER.
n = 4	DATA STACK FOR 4th LEVEL IN PAR. OPER
n = 5	DATA STACK FOR 5th LEVEL IN PAR. OPER
n = 6	RETURN VECTOR STACK (2)
n = 7	PROGRAM FLAG REGISTER (1)
n = 8	UNUSED IN VERSION 2 (RESERVED FOR F.BLK.) G.T.
n = 9	DEDICATED CONSTANT-STORAGE REGISTER

FUNCTION BLOCK

NOTES:

(1) -- FORMAT:

PT: 13 12 11 10 9 8 7 6 5 4 3 2 1 0

FLAG # LE - - - 0 1 2 3 4 5 6 7 8 9

(2) -- FORMAT FOR FULL STACK

16 BITS 10

PT: 13 12 11 10 9 8 7 6 5 4 3 2 1 0

RETURN VECTORS | | K 1st level | 2nd level | 3rd level |

(3) --- FORMAT FOR FULL STACK

PT: 13 12 11 10 9 8 7 6 5 4 3 2 1 0

OP. CODES-LEVELS | | 1st | 2nd | 3rd | 4th | 5th |

16 Bits ⇒ 4 BCD digits

INTERNAL FLAG VECTOR FORMAT

INTERNAL FLAG VECTOR, NORMALLY STORED IN M-REGISTER, CONTAINS MULTIVALUE FLAGS WHICH DEFINE MODE AND TYPE OF MACHINE OPERATION. INDIVIDUAL FLAGS ARE ACCESSIBLE VIA POINTER INSTRUCTIONS.

FORMAT:

<M>:

13	12	11	10	9	8	7	6	5	4	3	2	1	0

M13 ... DECIMAL POINT LOCATION (DPL) FLAG

0	...	FIX 0
1	...	FIX 1
2	...	FIX 2
3	...	FIX 3
4	...	FIX 4
5	...	FIX 5
6	...	FIX 6
7	...	UNUSED
8	...	
9	...	SCI. NOT

N . o o o o o o o o o o

M12 ... OPERATION MODE (OPM.) FLAG

0	...	MANUAL CONTROL
1	...	PROGRAM EXECUTION
2	...	UNUSED
3	...	
4	...	
5	...	
6	...	
7	...	
8	...	
9	...	

M11 ... DIGIT INTERPRETATION CODE

0	...	DATA ENTRY DIGIT
1	...	DEC. POINT LOCATION FLAG VALUE
2	...	I/O CHANNEL SELECT
3	...	PROGRAM FLAG NUMBER
4	...	RESERVED FOR FUNCTION BLOCK ROUTINES
5	...	
6	...	
7	...	
8	...	
9	...	

18 17 16 15 14 13 12 11 10 9

ADDRESS

		EERA
ISZ		
LIO	X_i	X_i ROM 0
SRI		X_i+1 ROM 1
ISI		X_i+2 ROM 2
RETURN		X_i+3 ROM 3
		X_i+4 ROM 4
		X_i+5 ROM 5
		X_i+6 ROM 6
		X_i+7 RETURN

0 → 9
1 → 8

CXM
PT6
P, CM1C
CXM
BRN

6-21-1972

M10 ... NUMBER OF ADJUSTMENT LEFT SHIFTS NEEDED FOR NEXT DIGIT TO BE ENTERED IN DATA ENTRY ROUTINE

M9 ... PARENTHESIS NESTING LEVEL INDICATOR

0 ... NO PARENTHESIS USED
1 ... ONE LEVEL DEEP
2 ... TWO LEVELS DEEP
3 ... THREE LEVELS DEEP
4 ... FOUR LEVELS DEEP
5 ... FIVE LEVELS DEEP
6 ...
7 ... } SYNTAX ERROR
8 ...
9 ...

M8 ... EQUAL KEY INTERPRETATION CODE

0 ... ARITHMETIC EQUAL OR SUBTOTAL
1 ... RESET MACHINE AND PRINT CLEAR
2 ...
3 ...
4 ...
5 ... } UNUSED
6 ...
7 ...
8 ...
9 ... PRINT TOTAL

M8

M7 ... ROUTINE USER CODE

0 ... MAIN UNIT
1 ... FUNCTION BLOCKS FIRST ROUTINE
2 ...
3 ...
4 ...
5 ... } RESERVED FOR USER OUTSIDE MAIN UNIT
6 ...
7 ...
8 ...
9 ...

M6 ... PROGRAM JUMP CONDITION FLAG

0 ... UNCONDITIONAL JUMP OR CONDITIONAL JUMP FOR CONDITION MET
1 ... CONDITIONAL JUMP FOR CONDITION NOT MET
2 ...
3 ...
: } UNUSED
9 ...

6-21-1972

M5 --- JSB LEVEL INDICATOR

0 --- NO SUBROUTINE CALLED
1 --- ONE LEVEL DEEP
2 --- TWO LEVELS DEEP
3 --- THREE LEVELS DEEP
4 ---
5 ---
6 ---
7 --- } SYNTAX ERROR
8 ---
9 ---

M4 --- D.S. ROUTINE USER FLAG ~~UNUSED IN VERSION 2~~

0 --- ACCMUL
1 --- ACCPLUS
2 --- ACCSUB
3 --- ACCDIV
4 --- ACCPOW
5 --- STORE
6 --- EXCHANGE
7 --- LISTDATA
8 ---
9 ---

M3;M2 TWO DIGIT OP. CODE OF THE OPERATION TO BE EXECUTED

M1, M0 TWO DIGIT OP. CODE OF THE CURRENTLY KEYED OPERATION

00 --- COPY C TO D-REG
01 --- ADD
02 --- SUB
03 --- MUL
04 --- DIV
05 --- XTY
06 --- STODIR
07 --- ACCPLUS
08 --- ACCSUB
09 --- ACCMUL
10 --- ACCDIV
11 --- ACCPOW
12 --- RECALL
13 --- RECADD
14 --- RECSUB
15 --- RECMUL
16 --- RECDIV
17 --- RECPOW
18 --- EXCH
19 ---
: } RESERVED FOR FUNCTION BLOCK FUNCTIONS
99 ---

6-22-1772

STATUS REGISTER USAGE

ST 0 ____> KEYDOWN FROM ANY KEYBOARD

ST 1 ____ AUXILIARY FLAG IN XTY ROUTINE INDICATES ODD INTEGER Y FOR NEGATIVE X;
JUMP TO SUBROUTINE FLAG;
RETURN ROUTINE FLAG;
(RESERVED FOR TRIG. FUNCTIONS);
SHIFT KEY ROUTINES AUX. FLAG FOR PRINT;

ST 2 ____ LEFT PARENTHESIS FLAG;
EXPO. ROUTINES COMMON FLAG;

ST 3 ____ "OPERATION"- SYNTAX FLAG;

ST 4 ____ JUMP RELATIVE FLAG;
AUXILIARY FLAG IN LEFT PAR. ROUTINE FOR IMPLY. MULT.
RIGHT PARENTHESIS AUX. FLAG.

ST 5 ____ DECIMAL POINT KEY FLAG;
COMMON FLAG FOR PROGRAM JUMPS;
SCI. NOTATION FORMAT FLAG;
DEC. LOG ROUTINE FLAG;
DO NOT PRINT FLAG;
AUXILIARY FLAG IN LEFT PAR. ROUTINE;
(RESERVED FOR TRIG. FUNCTIONS)

ST 6 ____ DECIMAL POINT SYNTAX FLAG;
STORE RESULT IN D.S. FLAG;

ST 7 ____ FIRST DATA DIGIT ENTRY SYNTAX FLAG;

ST 8 ____ RIGHT PARENTHESIS FLAG;
JSB TO LABEL AND GO TO LABEL FLAG;
XTY AND LOG. ROUTINES FLAG;

ST 9 ____ STOREX ROUTINE FLAG;
LABEL IN C-REG FLAG; (= COMPUTED LABEL FLAG);
CONDITION MET AUXILIARY FLAG;
SET-PROGRAM-FLAG FLAG;
SHIFT KEY ROUTINES AUXILIARY FLAG;
EXPONENTIAL ROUTINE FLAG;
STORE AND ACCUMULATE IN D.S. ROUTINES AUX. FLAG;
FORCED POINT FLAG;
(RESERVED FOR TRIG. ROUTINES)
AUX. FLAG IN LIST DATA ROUTINE

6-22-72

ST 10 ----- SHIFT KEY FLAG;
SECOND PRINT FLAG;
(RESERVED FOR TRIG. ROUTINES)

ST 11 ----- SINGLE OPERAND OPERATION FLAG;
AUXILIARY ERROR FLAG IN RIGHT PARENTHESIS ROUT;
DISPLAY ROUTINE FLAG;
RIGHT PARENTHESIS ROUTINE AUX. FLAG;
EXTERNAL DEVICE HARDWARE FLAGS:
- PRINTER HARDWARE ENABLE FLAG
- DATA ACCEPTED FOR PRINTER SECTOR FLAG;
- AUTO D.P. SWITCH FLAG;
- PRINT ON COMMAND SWITCH FLAG;
- FUNCTION BLOCK KEYBOARD FLAG;
- PROGRAM MEMORY ADDRESS NOT FOUND FLAG;
- DATA STORAGE REGISTER FOUND FLAG;

NOTE: " (RESERVED FOR TRIG. FUNCTIONS) MEANS THAT THESE FLAGS ARE USED IN DAVE'S TRIG. ROUTINES ONLY; THERE IS NO ASSURANCE THAT THERE WILL BE NO COLLISION WITH VERSION 2 SOFTWARE.

PRINT DRUM DESIGN

For Seiko Model 102 Digital Printer

T-REG C-REG

Sector # → / Column # ↓	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	13	12	11	10
0	$\bar{x}$	0	0	0	0	0	0	0	0	0	0	0	B	+	0	0	S	+
1	$\bar{y}$	1	1	1	1	1	1	1	1	1	1	1	C	1	1	1	1/x	x
2	(	2	2	2	2	2	2	2	2	2	2	2	D	2	2	2	(	=
3	$\bar{z}$	3	3	3	3	3	3	3	3	3	3	3	F	3	3	3	)	→
4	A	4	4	4	4	4	4	4	4	4	4	4	G	4	4	4	%	←
5	V	5	5	5	5	5	5	5	5	5	5	5	H	5	5	5	x/12	↑
6	$\mathcal{A}$	6	6	6	6	6	6	6	6	6	6	6	I	6	6	6	e^x	÷
7	#	7	7	7	7	7	7	7	7	7	7	7	J	L	7	7	L_n	−
8	r^2	8	8	8	8	8	8	8	8	8	8	8	M	G	8	8	L_g	⇄
9	−	9	9	9	9	9	9	9	9	9	9	9	A	−	9	9	L	#
10	T	S	.	T	P	.	.	.	.	.	.	.	T	T	.	D	I	S( )
11	R	E	R	R	E	R	R	T	D	A	T	A	E	R	E	B	R	T
12	C	C	N	M	M	E	N	P	R	G	M	E	N	D	B	A	"	S

1) This print drum configuration is only a modification of a previous design. See Seiko drawing number 86-22-17-1b.

2) All characters which were added or changed are shown in the accompanying drawings, except where the character was previously specified for the print drum.

3) The following positions are those which differ from the drums previously fabricated:

Sector 0

Column 18
Column 2

18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

ERROR DIS
X NOTE XX
COMMENT RED
PRGM BL
PER DATA DO
TERM. DATE ASS
CENTER RATE
RENTER PR AMT
STOP AGE
COMP CREDIT
ROM GO DEBIT
PERT BAL
ENTER CLEAR
ACNT END
NOM. GO TO
PEN RO IN
NO NO NO
GOOD

PRINT ROUTINE

```
LFPR1   PTØ             PREPARE SECTOR COUNTER
        IS2
        PRE             PRINTER ENABLE
        IS1
LFPR2   YS11            FLAG ?
        BRN   LFPR2     NO, WAIT
        RS11            YES, RESET FLAG
        IS2
        CCS             RIGHT PART OF PRINTER MASK
        TCS             LEFT  "  "  "  "
        IS1
        PLS             INCREMENT SECTOR COUNTER
        YP13            LAST SECTOR
        BRN   LFPR2     NO
LFPR3   YS11            FLAG ?
        BRN   LFPR3     NO, WAIT
        RS11            YES, RESET FLAG
        IS2
        ADV             PAPER ADVANCE
        IS1
```

DATA TO BE PRINTED SHOULD BE IN T&C REG

Doug Clifford

PROGRAMMING LANGUAGE FOR "5"

PRGM	NORM	END
IF	FLAG	LBLX
JMP	JSB	RET
LBL	STEP	T ()
INSERT	DELETE	LIST

--- To Enter the Programmer, use the shift key followed by "END" and "PRGM". This starts out at location 000 in the R/W memory.

To Enter at another address use "SHIFT", "JMP" followed by the address terminated in a "." . Then hit "PRGM" and the programming will begin at the specified state.

--- Programs are written using both the main unit keys and the function block keys. The program is written just as it would be executed in the normal mode, except for the branching statements.

The following instructions modify the execution of the next branch or return statement depending on whether the condition is met:

"IF","+"
"IF","-"
"IF","Ø"
"IF","LAST ENTRY"

} These two keystrokes are compacted into one single instruction.

"IF", "FLAG", "=" (= is any single digit , Ø-9)

"IF", "FLAG" is compacted into a single instruction which looks for a digit as the next instruction.

--- Labels allow the program to locate a certain step in R/W memory without knowing the absolute address. To program a Label use the "LBL" key followed by the label number terminated with a decimal point ".". The programming software compacts this sequence into three instructions.

1 - Label
2 - }
3 - } 3 digit label stored in the next two instructions

--- Branches - There are four branching instructions; 1. JMP Absolute, 2. JSB , 3. JMP LABEL, 4. JMP ± JMP LBLX is a modification of JMP LABEL. In all cases except JMP LBLX the branch is a 3 state instruction.

1. Branch type
2. Address, Label, or relative jump
3. compacted into 2 states.

The branches are all conditional. The "IF" statements, if met, force a skip of the next branch or return.

--- The Editing Mode is entered by a "SHIFT" "JMP" followed by the address of the state to be editted. The following keys can now be used.

INSERT - This sets the machine into the insert mode. Program steps are inserted in front of the state specified in the entry routine. The rest of the memory is moved down one state for everyone entered. The machine stays in the insert mode until a "NORM" key is depressed.

DELETE - This deletes the current program instruction and moves the rest of memory down one state. The only exception is when any of the 3 state instructions are deleted. All three states are deleted at once.

LIST - Lists out the program memory until an "END" statement is found.

NOTE: Do not try to replace or delete must the address of a three state branch. The whole instruction must be replaced.

--- STEP - The STEP key is the program mode steps through the program and lists out each state. In the normal mode "SHIFT", "STEP" allows a single step execution of the program.

--- CANCEL ENTRY - In the program mode this cancels the last step programmed or cancels the label or address being programmed. Repeated depressions of this key steps the program backwards and allows reprogramming of the states.

--- T ( ) - The table key contains a collection of seldom used routines.

T (Ø) = clear Data Storage
T (1) = list Data Storage
T (2) = check sum calculation
T (3) = next page instruction
T (4) = program paper advance
T (5) = Data storage decrement & test if Ø

ERROR Ø	Inconsistent sequence of keycodes or not in PRGM MODE
ERROR 1	"IF" followed by non zero digit
ERROR 2	Improper digit entry
ERROR 3	Combined keycode not completed
ERROR 4	Protected memory (cannot LIST, or EDIT)
ERROR 5	Jump to protected or illegal address
ERROR 6	Absolute address too large