FUNCTION	INSTRUCTION	MNEMONIC CODE	ADDRESSING MODES: NO *=N TAG	NO *=N CODE	MEMORY *=M TAG	MEMORY *=M CODE	INDIRECT *=I TAG	INDIRECT *=I CODE
	LOGICAL PRODUCT	LP*	0	020	n	021	n	022
	LOGICAL SUM	LS*	0	030	n	031	n	032
	LOAD A	LD*	0	100	n	101	n	102
ARITHMETIC	ADD TO A	AD*	0	120	n	121	n	122
&	SUBTRACT FROM A	SB*	0	130	n	131	n	132
LOGICAL	STORE A	ST*	-		n	201	n	202
	REPLACE ADD	RAM	-		n	221		
	REPLACE ADD ONE	RAO	-		n	231		
SHIFT	SHIFT A LEFT 1 BIT	SHA	0	001				
	JUMP DIRECT ZERO	ZJM			n	300		
TEST	" " NON-ZERO	NZM			n	301		
&	" " POSITIVE	PJM			n	302		
JUMP	" " NEGATIVE	NJM			n	303		
	" " UNCONDITIONAC	JMP			n	310		
	ERROR STOP		0	000				
SPECIAL	HALT	HLT	0	333				
	CLEAR INTERUPT LOCKOUT	CIL	0	023				
TAG MANIPULATION	A TO AUX. ADDRESS REGISTER	ATI	n		n	002		
	AUX. ADDRESS REGISTER TO A	ITA	n		n	003		
	A TO BER	ATE			n	010		
	A TO BXR	ATX			n	011		
INPUT	BER TO A	ETA			n	012		
&	INITIATE BUFFER INPUT	IBI			n	320		
OUTPUT	" " OUTPUT	IBO			n	321		
	" NORMAL INPUT	INP			n	322		
	" " OUTPUT	OUT			n	323		
	OUTPUT NO ADDRESS	ONA	0	330				
	INPUT TO A	INA	0	332				
	EXTERNAL FUNCTION	EXF	0	331				
	CCEAR BUFFER CONTROLS	CBC	0	013				

Telegrammer Punctic Codes

LOAD T/P VIA TYPEWRITER IN QUARTIC DIGITS

1) SET ONE TAB ON TYPEWRITER

2) LOAD PAPER TAPE PROGRAM
   A) FOR 2K MEMORY LOAD STARTING IN 030000
   B) FOR 4K MEMORY LOAD STARTING IN 330000, AND ENTER 0033 INTO 330102

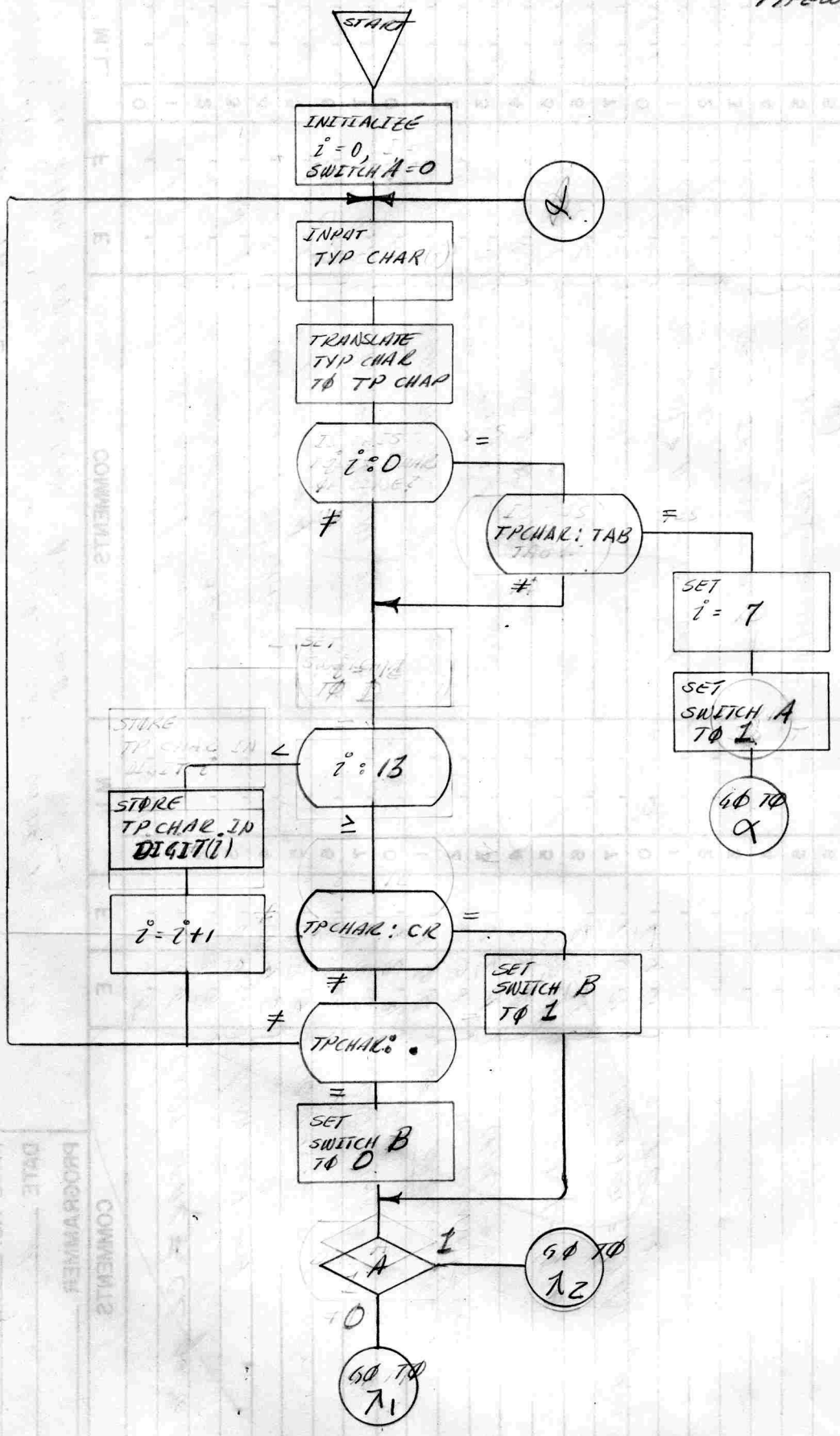
3) RUN PROGRAM
   A) FOR 2K MEMORY START IN 030101
   B) FOR 4K MEMORY START IN 330101

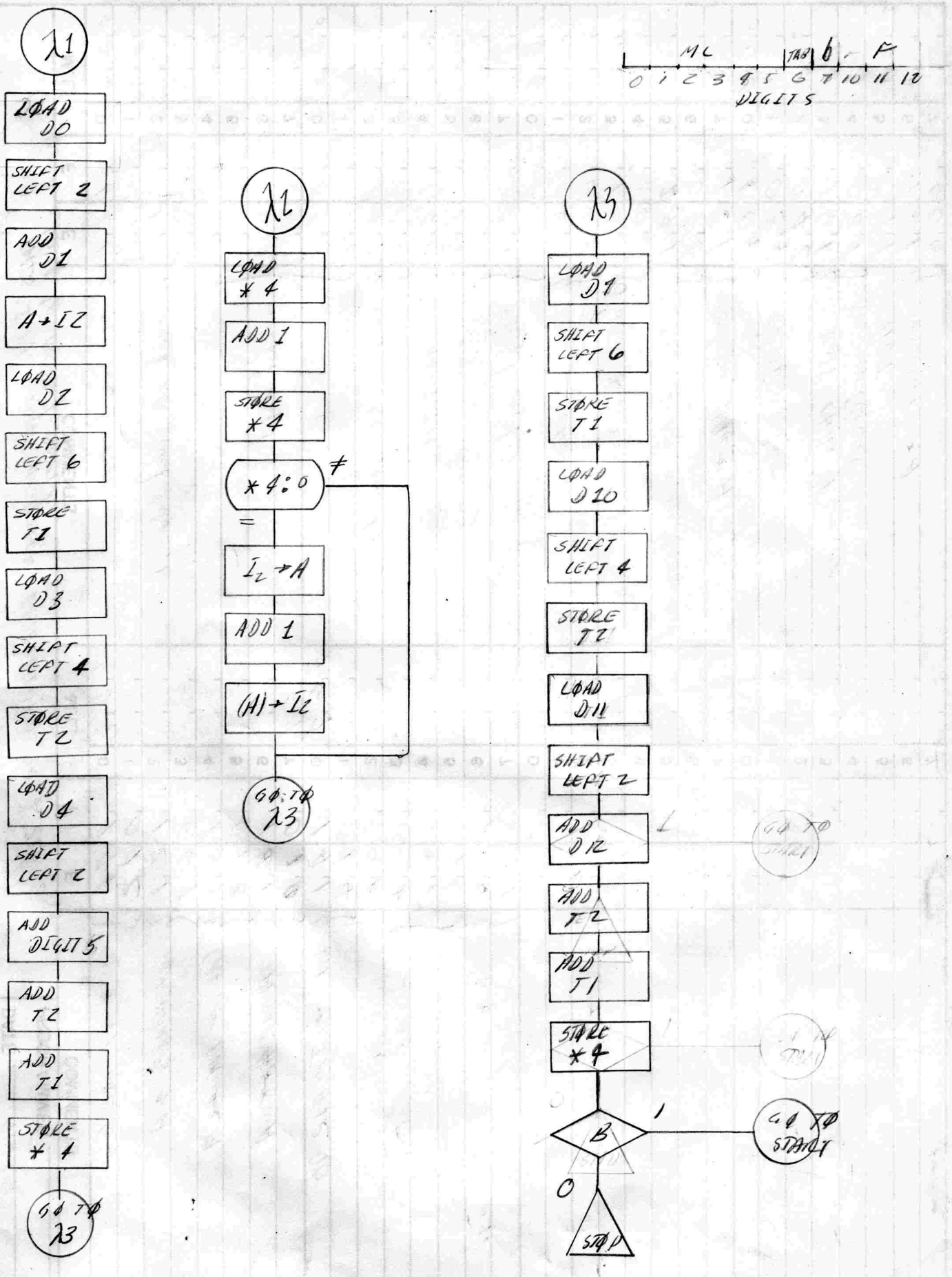
4) TO LOAD VIA TYPEWRITER
   A) RUN
   B) TYPE THE 6 DIGIT ML OF THE WORD TO BE LOADED
   C) TAB
   D) TYPE THE 4 DIGIT WORD TO BE LOADED
   E) CARRIAGE RETURN - THIS CAUSES THE WORD TO BE LOADED INTO THE ML SPECIFIED.
   F) FOR ANY FOLLOWING WORDS WHICH ARE TO BE LOAD IN SEQUENTIAL ML'S REPEAT STEP 4-C, D AND E
   G) FOR FOLLOWING WORDS WHICH WILL BREAK THE SEQUENCE OF ML'S PERFORM 4-B, C, D AND E

5) END OF LOADING IS ACCOMPLISHED WHEN A PERIODE IS TYPED IN THE PLACE OF A CARRIAGE RETURN. THE CONSOLE WILL DISPLAY 0333 IN A, AND 033003 OR 333003 IN P.

6) ERRORS IN TYPING MAY BE CORRECTED BY SPACING OR TYPING CHARACTERS TO THE CARRIAGE RETURN SPACE DO A CARRIAGE RETURN AND PERFORMING STEPS 4-B, C, D AND E.

LOAD T/P VIA TYPEWRITER




λ1
LOAD D0
SHIFT LEFT 2
ADD D1
A → I2
LOAD D2
SHIFT LEFT 6
STORE T1
LOAD D3
SHIFT LEFT 4
STORE T2
LOAD D4
SHIFT LEFT 2
ADD DIGITS
ADD T2
ADD T1
STORE *4
GO TO λ3
λ2
LOAD *4
ADD 1
STORE *4
*4 : 0
≠
=
I2 → A
ADD 1
(A) → I2
GO TO λ3
λ3
LOAD D1
SHIFT LEFT 6
STORE T1
LOAD D10
SHIFT LEFT 4
STORE T2
LOAD D11
SHIFT LEFT 2
ADD D12
ADD T2
ADD T1
STORE *4
B
1
0
STOP
MC
TAG
6
P
0 1 2 3 4 5 6 7 10 11 12
DIGITS

	ML	T	F	
K	3 0 0 0 0			TEMP 1
	1			TEMP 2
	2			SWITCH a
	3			SWITCH b
	0 0 1 0			TP CHAR
	1			COUNT
	2			DIGIT 0
	3			" 1
	0 0 2 0			" 2
	1			" 3
	2			" 4
	3			" 5
	0 0 3 0			" 6
	1			" 7
	2			" 10
	3			" 11
	0 1 0 0			" 12
I	1	0 1 0 0		3 → A
	2	0 0 0 3		
	3	1 0 0 2		(A) → AAR 1 AAR1 is the auxiliary address register used in the execution of this program
	0 1 1 0	0 1 0 0		
	1	0 0 0 0		initialize COUNT at -0-
	2	1 2 0 1		
	3	0 0 1 1		
	0 1 2 0	0 1 0 0		
	1	0 0 0 0		set switch a to -0- this is a constant
	2	1 2 0 1		
	3	0 0 0 2		
	0 1 3 0	0 1 0 0		
	1	0 0 1 2		initialize ML of first DIGIT to be stored
	2	1 2 0 1		
	3	1 1 0 3		
X	0 2 0 0	0 3 3 1		
	1	0 2 0 2		request TYP status
	2	0 2 0 0		
	3	0 3 3 2		status response → A
	0 2 1 0	1 3 0 1		non-zero go to 2
	1	0 2 0 0		
	2	0 3 3 1		
	3	0 2 0 2		request TYP input

	ML	T	F	
	3 0 2 2 0	0	0 2 0	
		1	0 3 3 2	TYPWRTR CHAR → A
		2	1 3 0 0	zero go to α } this is redundant
		3	0 2 0 0	
	0 2 3 0	0	0 1 2 0	add ML of first entry of TP CHAR → TP CHAR
		1	3 0 0 0	translation table
		2	1 2 0 1	store operand of translation instruction
		3	0 3 0 1	
	0 3 0 0	1	1 0 1	translation → A
		1	3 3 3 3	
		2	1 2 0 1	(A) → TP CHAR
		3	0 0 1 0	
	0 3 1 0	1	1 0 1	COUNT → A
		1	0 0 1 1	
		2	1 3 0 1	non zero go to β check for end of inst.
X		3	1 0 3 0	
	0 3 2 0	1	1 0 1	TP CHAR → A
		1	0 0 1 0	
		2	0 1 3 0	A - 51/8 (TAB)
		3	0 2 2 1	
	0 3 3 0	1	3 0 1	non zero go to β check for end of inst.
X		1	1 0 3 0	
		2	0 1 0 0	1 → A
		3	0 0 1 3	
	1 0 0 0	1	2 2 1	COUNT = 1
		1	0 0 1 1	
X		2	0 1 0 0	set operand of store in appropriate DIGIT
		3	0 0 1 3	to 0031
	1 0 1 0	1	2 2 1	
		1	1 1 0 3	
		2	0 1 0 0	set switch a to 1
		3	0 0 0 1	
	1 0 2 0	1	2 0 1	
		1	0 0 0 2	
		2	1 3 1 0	go to α
		3	0 2 0 0	
β	1 0 3 0	0	1 3 0	A - 13/8 → A
		1	0 0 2 3	
		2	1 3 0 2	positive go to δ check last TP CHAR of typed
X		3	1 1 2 2	line (COUNT ≥ 13)

3	1	1	0	0	TP CHAR → A
			1	0	
			2	1	
			3	3	
	1	1	1	0	replace add no to operand of store in appropriate
			1	1	DIGIT instruction
			2	1	COUNT+1 → COUNT
			3	0	
	1	1	2	0	go to α
			1	0	
γ			2	1	TP CHAR → A
			3	0	
	1	1	3	0	A - 45/8 (CR)
			1	0	
			2	1	zero go to δ set switch b to 1
			3	1	
	1	2	0	0	A - 45/8 + 3/8 ≃ A - 42/8 (.)
			1	0	
			2	1	non zero go to α
			3	0	
	1	2	1	0	
			1	0	set switch b to 0
			2	1	
			3	0	
	1	2	2	0	go to δ switch a text
			1	1	
δ			2	0	
			3	0	set switch b to 1
	1	2	3	0	
			1	0	
ε			2	1	switch a → A
			3	0	
	1	3	0	0	non zero go to λ2
			1	2	
λ1			2	1	DIGIT 0 → A
			3	0	
	1	3	1	0	shift left 2 bits
			1	0	
			2	1	add DIGIT 1
			3	0	

3	1	3	2	0	2	0	0	2	(A) → AAR 2
				1	1	1	0	1	DIGIT 2 → A
				2	0	0	2	0	
				3	0	0	0	1	
		1	3	3	0	0	0	1	
				1	0	0	0	1	shift left 6 bits
				2	0	0	0	1	
				3	0	0	0	1	
	2	0	0	0	0	0	0	1	
				1	1	2	0	1	store TEMP1
				2	0	0	0	0	
				3	1	1	0	1	DIGIT 3 → A
	2	0	1	0	0	0	2	1	
				1	0	0	0	1	
				2	0	0	0	1	shift left 4 bit
				3	0	0	0	1	
	2	0	2	0	0	0	0	1	
				1	1	2	0	1	store TEMP2
				2	0	0	0	1	
				3	1	1	0	1	DIGIT 4 → A
	2	0	3	0	0	0	2	2	
				1	0	0	0	1	shift left 2 bits
				2	0	0	0	1	
				3	1	1	2	1	add DIGIT 5
	2	1	0	0	0	0	2	3	
				1	1	1	2	1	add TEMP2
				2	0	0	0	1	
				3	1	1	2	1	add TEMP1
	2	1	1	0	0	0	0	0	
				1	1	2	0	1	store in ML which contains ML of this
				2	2	3	3	2	tipped prog step *4
				3	1	3	1	0	go to λ3
	2	1	2	0	2	2	0	1	
				1	1	2	3	1	*4 +1 → *4
				2	2	3	3	2	
				3	1	3	0	1	non zero go to λ3
	2	1	3	0	2	2	0	1	
				1	2	0	0	3	AAR 2 → A
				2	0	1	2	0	A+1 → A
				3	0	0	0	1	

	3	2	2	0	0	2	0	0	2	(A) → AARZ
13					1	1	1	0	1	DIGIT 4 → A
					2	0	0	3	1	
					3	0	0	0	1	
		2	2	1	0	0	0	0	1	
					1	0	0	0	1	shift left 6 bits
					2	0	0	0	1	
					3	0	0	0	1	
		2	2	2	0	0	0	0	1	
					1	1	2	0	1	store TEMP1
					2	0	0	0	0	
					3	1	1	0	1	DIGIT 10 → A
		2	2	3	0	0	0	3	2	
					1	0	0	0	1	
					2	0	0	0	1	shift left 4 bits
					3	0	0	0	1	
		2	3	0	0	0	0	0	1	
					1	1	2	0	1	store TEMP2
					2	0	0	0	1	
					3	1	1	0	1	DIGIT 11 → A
		2	3	1	0	0	0	3	3	
					1	0	0	0	1	shift left 2 bits
					2	0	0	0	1	
					3	1	1	2	1	add DIGIT 12
		2	3	2	0	0	1	0	0	
					1	1	1	2	1	add TEMP2
					2	0	0	0	1	
					3	1	1	2	1	add TEMP1
		2	3	3	0	0	0	0	0	
					1	2	2	0	1	store instruction in ML determined in 12 or
14					2	0	0	0	0	13
					3	1	1	0	1	switch b → A
		3	0	0	0	0	0	0	3	
					1	1	3	0	1	sum zero go to initialize
					2	0	1	1	0	
					3	0	3	3	3	HLT
		3	0	1	0					
					1					
					2					
					3					

```
3 3 0 2 0
        1
        2
        3
  3 0 3 0
        1
        2
        3
  3 1 0 0
        1
        2
        3
  3 1 1 0
        1
        2
        3
  3 1 2 0
        1
        2
        3
  3 1 3 0
        1
        2
        3
  3 2 0 0
        1
        2 0 2 0 2        42
        3
  3 2 1 0
        1 0 2 1 1        45  CR
        2
        3
  3 2 2 0
        1 0 2 2 1        51  TAB
        2
        3
  3 2 3 0
        1
        2 0 0 0 0        56  0
        3
```

3	3	3	0	0	0	0	1	3	60	1
				1						
				2	0	0	1	0	62	7
				3						
	3	3	1	0	0	0	0	3	64	3
				1						
				2	0	0	1	1	66	5
				3						
	3	3	2	0	0	0	0	2	70	2
				1						
				2	0	0	1	2	72	6
				3						
	3	3	3	0	0	0	0	1	74	1
				1						
				2						
				3						

				0						
				1						
				2						
				3						
				0						
				1						
				2						
				3						
				0						
				1						
				2						
				3						
				0						
				1						
				2						
				3						
				0						
				1						
				2						
				3						

DUMP T/P MEMORY VIA TYPEWRITER IN QUARTIC DIGITS

1) SET ONE TAB ON TYPEWRITER AND PERFORM CARRIAGE RETURN

2) LOAD PAPER TAPE PROGRAM:
   A) FOR 2K MEMORY
      1) FIRST WORD ADDRESS TAG INTO 031300
      2) " " " OPERAND " 031301
      3) LAST " " TAG " 031302
      4) " " " OPERAND " 031303
   B) FOR 4K MEMORY
      1) FIRT WORD ADDRESS TAG INTO 331300
      2) " " " OPERAND " 331301
      3) LAST " " TAG " 331302
      4) " " " OPERAND " 331303
      5) ENTER 0033 INTO 331311

3) RUN PROGRAM
   A) FOR 2K MEMORY START IN 031310 AND RUN
   B) " " " " " 331310 " "

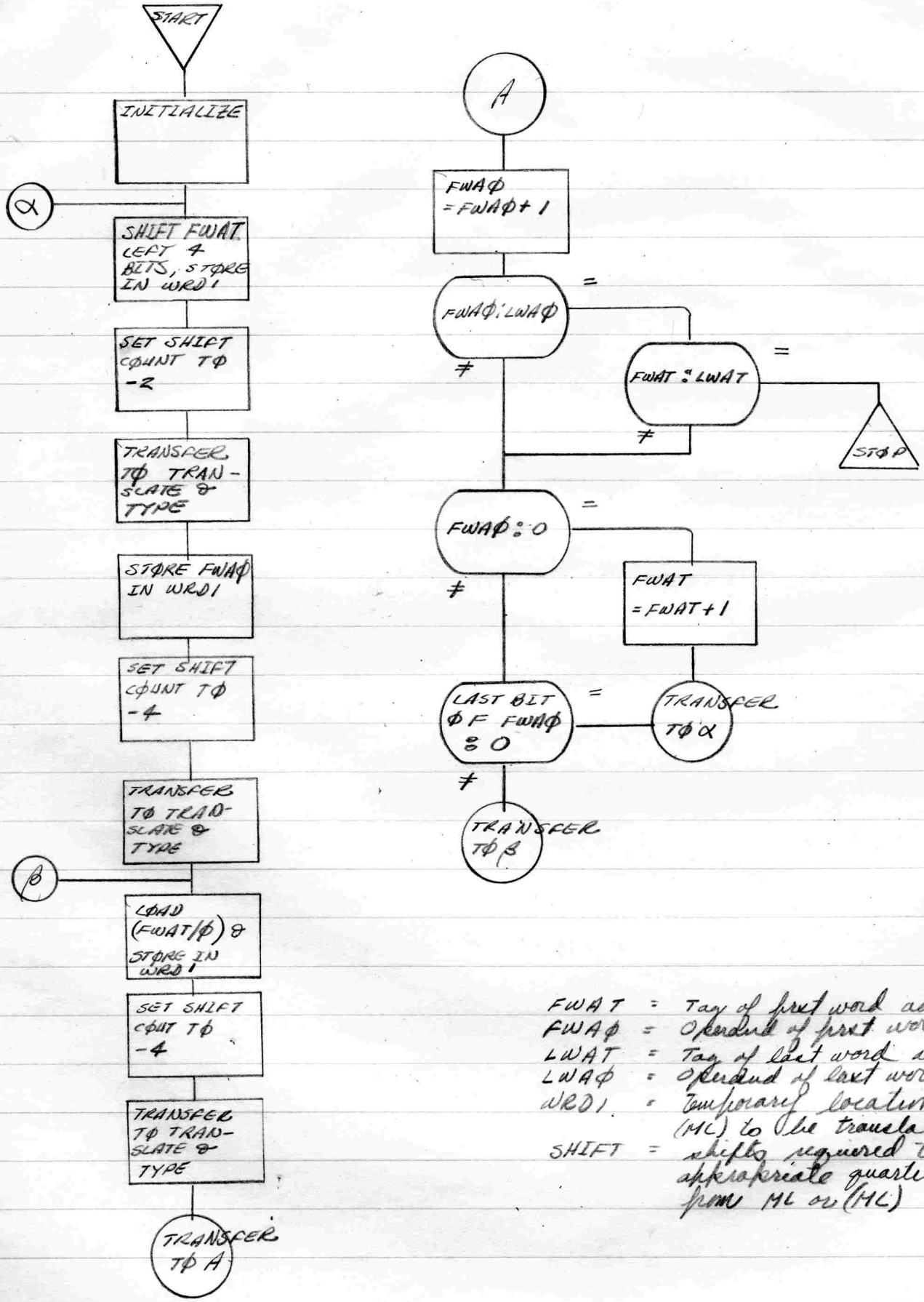
4) END OF DUMP IS INDICATED WHEN THE CONSOLE DISPLAYS 0333 IN A, AND 033130 OR 333130 IN P

7) TYPING WILL BE CONTINUOUS THEREFORE CONTINUOUS FORM PAPER SHOULD BE USED IN THE TYPEWRITER OR THE TELEPROGRAMER MUST BE TAKEN OUT OF RUN AT THE BOTTOM OF SHEET AND A NEW SHEET INSERTED AND THE TELEPROGRAMER PLACED IN RUN.

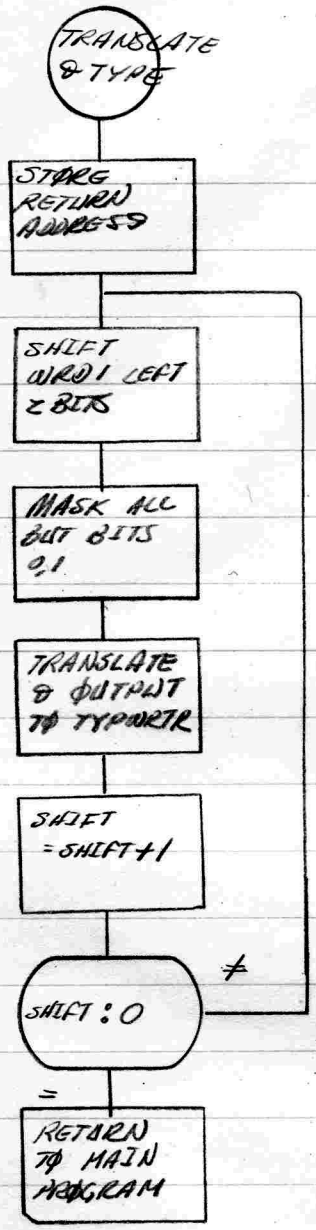
ML (ML)

031310	0100
	0003
	1002
	1101
031320	1300
	0001
	0001
	0001
031330	0001
	1201
	1233
	0100
032000	3331
	1201
	1232
	0100
032010	2013
	1310
	3131
	1101
032020	1301
	1201
	1233
	0100
032030	3323
	1201
	1232
	0100

DUMP T/P MEMORY VIA TYPEWRITER



FWAT = Tag of first word address
FWAØ = Ordinal of first word address
LWAT = Tag of last word address
LWAØ = Ordinal of last word address
WRD1 = Temporary location of ML or (ML) to be translated
SHIFT = shifts required to extract appropriate quartic digits from ML or (ML)


TRANSLATE & TYPE
STORE RETURN ADDRESS
SHIFT WORD 1 LEFT 2 BITS
MASK ALL BUT BITS 0,1
TRANSLATE & OUTPUT TO TYPWRTR
SHIFT = SHIFT+1
SHIFT : 0
≠
=
RETURN TO MAIN PROGRAM

page constants

	0			
	1			
	2			
	3			
	0			
	1			
	2			
	3			
	0			
	1			
	2			
	3			
121	0	0232		-0-
	1	0330		1
	2	0320		2
	3	0310		3
122	0			
	1			
	2			
	3			
123	0			
	1			
	2			SHFT
	3			WORD
130	0			FWAT
	1			FWAØ
	2			LWAT
	3			LWAØ

```
                1310 0  0100   LDN
                     1  0003
                     2  0002   ATI
A                    3  1101   LDM
A           1320 0  1300   FWAT
                     1  0001   SHA
                     2  0001    "
                     3  0001    "
            1330 0  0001    "
                     1  1201   STM
                     2  1233   WORD
                     3  0100   LDN
            2000 0  3331
                     1  1201   STM
                     2  1232   SHFT
                     3  0100   LDN
            2010 0  2013   *1
                     1  1310   JMP
                     2  3131   TYPE
*1                   3  1101   LDM
            2020 0  1301   FWA0
                     1  1201   STM
                     2  1233   WORD
                     3  0100   LDN
            2030 0  3323
                     1  1201   STM
                     2  1232   SHFT
                     3  0100   LDN
```

	2100	2103	*2	
	1	1310	JMP	
	2	3131	TYPE	
*2	3	0100	LDN	
	2110	0221	TAB	
	1	1201	STM	
	2	3321	*11	
	3	0100	LDN	
	2120	3332		
	1	1201	STM	
	2	1233	SHFT	
	3	0100	LDN	
	2130	2201	*3	
	1	1201	STM	
	2	3333	RTRN	
	3	1310	JMP	
	2200	3233	AGAIN	
*3	1	1101	LDM	
	2	1300	FWAT	
	3	2002	ATI	
	2210	1101	LDM	
	1	1301	FWAΦ	
	2	1201	STM	
	3	2221	*9	
	2220	2101	LDM	
*4	1	0000		
	2	1201	STM	
	3	1233	WORD	

		2	2	3	0	0	1	0	0	LDN
					1	3	3	2	3	
					2	1	2	0	1	STM
					3	1	2	3	2	SHFT
			2	3	0	0	1	0	0	LDN
					1	2	3	1	0	*5
					2	1	3	1	0	JMP
					3	3	1	3	1	TYPE
*5			2	3	1	0	0	1	0	0	LDN
					1	0	2	1	1	CR
					2	1	2	0	1	STM
					3	3	3	2	1	*11
			2	3	2	0	1	0	0	LDN
					1	3	3	3	2	
					2	1	2	0	1	STM
					3	1	2	3	2	SHFT
			2	3	3	0	1	0	0	LDN
					1	3	0	0	2	*6
					2	1	2	0	1	STM
					3	3	3	3	3	RTRN
		3	0	0	0	1	3	1	0	JMP
					1	3	2	3	3	AGAIN
*6					2	1	2	3	1	RAΦ
					3	1	3	0	1	FWAΦ
		3	0	1	0	1	1	3	1	SBM
					1	1	3	0	3	LWAΦ
					2	1	3	0	1	NZM
					3	3	0	3	2	*7

```
                3020  1  1101   LDM
                      1  1300   FWAT
                      2  1131   SBM
                      3  1302   LWAT
                3030  1  1300   ZJM
                      1  3 180  END
*1                    2  1101   LDM
                      3  1301   FWAØ
                3100  1  1300   ZJM
                      1  3120   *6
                      2  0020    LPN
                      3  0003
                3110  1  1301   NEM
                      1  2103   *2
                      2  1 3 10  JPM
                      3  1 3 13   A
*8              3120  1  2311   RAØ
                      1  1300   FWAT
                      2  1 3 10  JMP
                      3  1 3 13   A
END             3130  0  0333   HLT
TYPE                  1  1201   STM
                      2  3333   RTRN
*9                    3  1101   LDM
                3200  1  2333   WORD
                      1  0001   SHA
                      2  0001   SHA
                      3  1201   STM
```

		3	2	1	0	1	2	3	3	WORD	
					1	0	0	2	0	LPN	
					2	0	0	0	3		
					3	0	1	2	0	ADN	
		3	2	2	0	1	2	1	0	TBL	
					1	1	2	0	1	STM	
					2	3	2	3	0	* 10	
					3	1	1	0	1	LDM	
* 10		3	2	3	0	0	0	0	0		
					1	1	2	0	1	STM	
					2	3	3	2	1	* 11	
AGAIN					3	0	3	3	1	EXF	
		3	3	0	0	0	2	0	2		} T/W STATUS
					1	0	2	0	0		}
					2	0	3	3	2	INA	
					3	1	3	0	1	NEM	
		3	3	1	0	3	2	3	3	AGAIN	
					1	0	3	3	1	EXF	
					2	0	2	0	2		} T/W OUTPUT
					3	0	0	2	0		}
		3	3	2	0	0	3	3	0	ØNA	
* 11					1	0	0	0	0		
					2	1	2	3	1	RAØ	
					3	1	2	3	2	SHFT	
		3	3	3	0	1	3	0	1	NEM	
					1	3	1	3	3	* 9	
					2	1	3	1	0	JMP	
RTRN					3	0	0	0	0		

STORAGE OF BLOCK TO BE TRANSMITTED

DEC.	(ML)	OCTAL	QUARTIC
000	SYNCH 1	000	
1	SYNCH 2	1	
2	TYPE	2	
3	COUNT	3	
4		4	
5		5	
6		6	
7		7	
8	CHARACTER [1]	10	
9	" [2]	11	
10	" [3]	12	
11			
12			
13			
14			
15			
16			
17			
243			
244			
245			
246			
247	CHARACTER [240]	361	
248	EDC 1	370	
249	" 2	371	
250	" 3	372	
251	" 4	373	
252	" 5	374	
253	" 6	375	
254	" 7	376	
255	" 8	377	
256			

SEND & RCV ROUTINES

SYMBOL	OCTAL ML	T	F/E	MNEMONIC	QUARTIC T	F/E	ML	t	
SEND	0000	0	20	0 LDN			000000	2	set aux addr. Reg. 2
	1	0	00	0			01		for storage block ML's
	2	2	02	2 ATI			02	1	
	3	0	20	0 LDN			03	2	initialize to out
	4	0	00	0			10		put first CHARACTER from
	5	1	41	1 STM			11	3	storage block
	6	0	17	*1			12		
	7	0	15	0 EXF			13	3	
*1	10	0	36	036			20		put DCS in send mode
	11	0	01	001			21		
	12	0	20	0 LDN			22	2	
	13	0	00	0			23		set B = 0
	14	1	41	1 STM			30	3	
	15	2	45	B			31		
*1	16	2	21	2 LDM			32	3	(CHARACTER[C]) → A
*1	17	0	00	000			33		
	20	1	41	1 STM			000100	3	store in operand of
	21	0	23	*2			01		ØNA
	22	0	14	0 ØNA			02	2	output one word
*2	23	0	00	000			03		instruction
	24	0	01	0 SHH			10	1	
*2	25	0	01	0 SHA			11	1	
	26	0	01	0 SHH			12	1	
	27	0	01	0 SHA			13	1	shift WORD left 7 bit
	30	0	01	0 SHH			20	1	(same shift WORD right
	31	0	01	0 SHA			21	1	1 bit)
	32	0	01	0 SHA			22	1	
	33	1	41	1 STM			23	3	store in operand of ØNA
	34	0	23	*2			30		instr.
	35	1	51	1 RAØ			31	4	increase B by 1
	36	2	45	B			32		
	37	0	34	0 SBN			33	2	

SEND & RCV ROUTINES
PAGE: 2

SYMBOL	OCTAL MC	T	F/E	MNEMONIC	QUARTIC T	F/E	MC	t	
	0040	0	10	8			000200		subtract 8 from B
	41	1	63	/ NJM			01	2	B<8 go to output shifted
NEW SEND	42	0	22	*2-1			02		word
	43	1	51	/ RAØ			03	4	increase operand of load
	44	0	17	*1			10		CHARACTER instruction
	45	1	64	/ ZJM			11	2	CHARACTER MC = 0 go
	46	0	63	RCV			12		to receive routine (RCV)
	47	2	35	2 SBM			13	3	subtract 1 from operand of
	50			COUNT			20		load CHARACTER instr. words to be trans +1
	51	1	63	/ NJM			21	2	more wds to be sent go to
	52	0	16	*1-1			22		load CHARACTER
	53	1	21	/ LDM			23	3	
	54	3	70	2Ø			30		set operand of load CHARACTER
	55	1	41	/ STM			31	3	instr to EDC1 (Error Detection
	56	0	17	*1			32		Code)
	57	1	64	/ JMP			33	2	go to load CHARACTER
	60	0	16	*1-1			000300		routine
	61						01	2	
	62						02		
RCV	63	0	20	0 LDN			03	2	
	64	0	00	0			10		
	65	1	41	/ STM			11	3	put 0 in ALPHA, BETA
	66	2	46	WORD			12		& WORD
	67	1	41	/ STM			13	3	
	70	2	47	ALPHA			20		
	71	1	41	/ STM			21	3	
	72	2	50	BETA			22		
	73	0	75	0 EXF			23	3	
	74	0	36	36			30		set DCH in receive mode
	75	0	02	02			31		
IDLE	76	0	13	0 CIL			32	1	clear interrupt lock out
	77	1	02	/ ATI			33	1	idle lock

SYMBOL	OCTAL MC	T	F/E	MNEMONIC	QUARTIC T	F/E	MC	L	
	0100	1	64	1 JMP			001000	2	idle loop
	01	0	76	IDLE			01		
α	02	1	21	1 LDM			02	3	
	03	2	47	ALPHA			03		
	04	1	60	1 EJM			10	2	
	05	1	17	2 α0			11		
	06	0	34	0 SBN			12	2	go to various RCV
	07	0	01	1			13		routines depending on
	10	1	60	1 EJM			20	2	value of ALPHA
	11	1	37	2 α1			21		
	12	0	34	0 SBN			22	2	
	13	0	01	1			23		
	14	1	60	1 EJM			30	2	
	15	1	47	2 α2			31		
	16	0	77	0 HLT			32	1	
α	17	0	76	0 INA			33	2	add input bit to
	20	1	51	1 RAD			001100	4	low order of WORD
	21	2	46	WORD			01		
	22	2	35	2 SBM			02	3	if WORD = SYNCH go to
	23	0	00	SYNCH			03		set ALPHA to 1
	24	1	60	1 EJM			10	2	
	25	2	37	SETα1			11		
	26	1	21	1 LDM			12	3	
	27	2	46	WORD			13		
	30	0	10	0 LPN			20	2	
	31	1	77	177$_8$			21		if WORD ≠ SYNCH put zeros
	32	0	01	0 SHA			22	1	in high order of WORD & shift
	33	1	41	1 STM			23	3	left 1, and go to idle
	34	2	46	WORD			30		routine.
	35	1	64	1 JMP			31	2	
	36	0	76	IDLE			32		
α1	37	0	20	0 LDN			33	2	

SEND & RCV ROUTINES

SYMBOL	OCTAL MC	T	F/E	MNEMONIC	QUALTIC T	F/E	MC	L	
	0140	0	00	0			00 1200		
	41	1	41	/ STM			01	3	put -0- in B & WORD
	42	2	45	B			02		idle routine.
	43	1	41	/ STM			03	3	
	44	2	46	WORD			10		
	~~45~~	1	55	/ RAØ			11	4	set ALPHA to 2
	46	2	47	ALPHA			12		
α2	47	0	76	0 INA			13	2	input a bit and
	50	1	51	/ RAM			20	4	add to lower order pos.
	51	2	46	WORD			21		of WORD
	52	1	55	/ RAØ			22	4	increment B by 1
	53	2	45	B			23		
	54	0	34	0 SBN			30	2	if B=8 (ie 7 bits have been
	55	0	08	8			31		added WORD) go to β
	56	1	62	/ PJM			32	2	routine
	57	1	67	β			33		
	60	1	21	/ LDM			00 1300	3	if B<8 (ie less than 7 bits
	61	2	46	WORD			01		have been added WORD)
	62	0	01	0 SHA			02	1	shift WORD left one bit
	63	1	41	/ STM			03	3	add go to Idle routine
	64	2	46	WORD			10		
	65	1	64	/ JMP			11	2	
β	66	0	76	IDLE			12		
	67	1	21	/ LDM			13	3	
	70	2	50	BETA			20		if WORD = SYNCH go to
	71	2	35	2 SBM			21	3	Set α1 routine
	72	0	00	/ SYNCH			22		
	73	1	60	/ EJM			23	2	
	74	2	37	/ SETα1			30		
	75	1	55	/ RAØ			31	4	set BETA to 1
	76	2	50	BETA			32		
	77	0	20	0 LDN			33	2	

SYMBOL	OCTAL ML	T	F/E	MNEMONIC	QUARTIC T	F/E	ML	t	
	0200	0	02	/ Z			002000		set operand of store
	01	1	41	/ STM			01	3	WORD instruction to
	02	2	06	*5			02		initial value.
B1	03	1	21	/ LDM			03	3	
	04	2	46	WORD			10		store WORD in receive
	05	1	41	/ STM			11	3	T/P block.
*5	06	0	00	000			12		
	07	1	55	/ RAΦ			13	1	increment operand of store
	10	2	06	*5			20		WORD instruction by 1
	11	1	60	/ ZJM			21	2	if operand = 0 (ie block is
	12			CALCULATE RCV EDC			22		full) go to calculate EDC
	13	0	34	0 SBN			23	2	
	14	0	08	8			30		if operand < 8 (ie header has
B5	15	1	63	/ NJM			31	2	not been completed yet) go
	16	2	37	/ SETα1			32		to set α1 routine
	17	1	21	/ LDM			33	3	
	20	2	06	/ *5			002100		if operand ≥ 5 but < COUNT
	21	2	35	2 SBM			01	3	(ie block of words to be input
	22			COUNT			02		has not been completed yet)
	23	1	63	/ NJM			03	2	go to set α1 routine.
	24	2	37	/ SETα1			10		
	25	1	21	/ LDM			11	3	
	26	2	06	/ *5			12		if operand ≥ COUNT and ≥ 298
B6	27	0	34	0 SBN			13	2	(ie EDC's are being input)
	30	3	70	298			20		go to set α1 routine
	31	1	62	/ RJM			21	2	
	32	2	37	SETα1			22		
	33	0	20	0 LDN			23	2	if operand ≥ COUNT but < 298
	34	3	70	298			30		(ie WORDS have been input
	35	1	41	/ STM			31	3	but EDC's remain to
	36	2	06	*5			32		be input) set operand
SETα1	37	0	20	0 LDN			33	2	to 298 & go to set α1

	OCTAL				QUARTIC			
SYMBOL	ML	T	F/E	MNEMONIC	T	F/E	ML	E
	0240	0	01	1			00220	0
	41	1	41	1 STM			01	3
	42	2	47	ALPHA			02	
	43	1	64	1 JMP			03	2
	44	0	76	IDLE			10	
B	45	0	00	B			11	
WORD	46	0	00	WORD			12	
ALPHA	47	0	00	ALPHA			13	
BETA	50	0	00	BETA			20	
	51						21	
	52						22	
	53						23	
	54						30	
	55						31	
	56						32	
	57						33	
	60						00230	0
	61						01	
	62						02	
	63						03	
	64						10	
	65						11	
	66						12	
	67						13	
	70						20	
	71						21	
	72						22	
	73						23	
	74						30	
	75						31	
	76						32	
	77						33	

set ALPHA to 7 and go to Idle routines