Delete near end of program. If the delete command inadvertently comprises more program words than contained in the section to be deleted at the end of a program, words at the beginning of the program may disappear as well (*cyclic* delete action). Moreover, if the program has only one NOP instruction – i.e. on the last line, as prescribed above – no NOP words will be left. So, care should be exercized here; see Fig. 4-7.

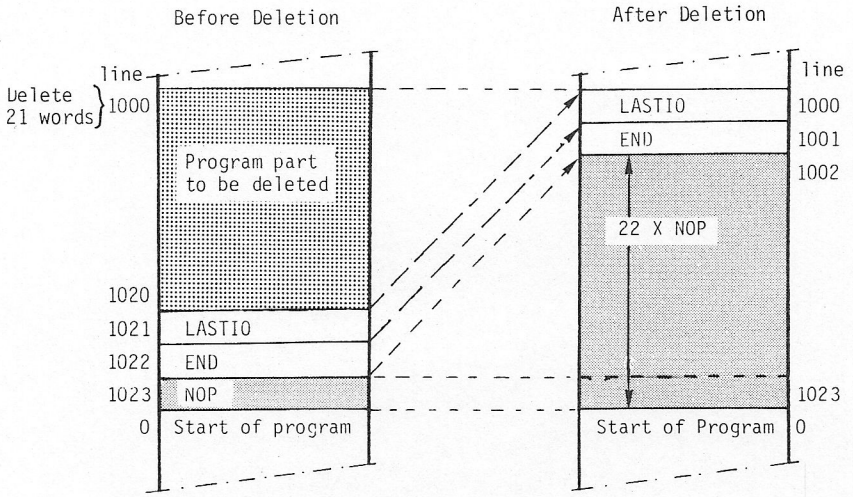


Fig. 4-7 Not more than 21 lines must be deleted in this example.

Modification of JUMP instructions. Because program words are shifted during deletion, it may become necessary to adapt JUMP instructions. Fig. 4-8 illustrates one example. If the relative jump JFRF is not adapted, the logic instruction 'AND 30.0' will not be seen. An absolute jump should be modified here in a similar way. Jumps are found using the "Search instruction" facility, Section 4.3.6.

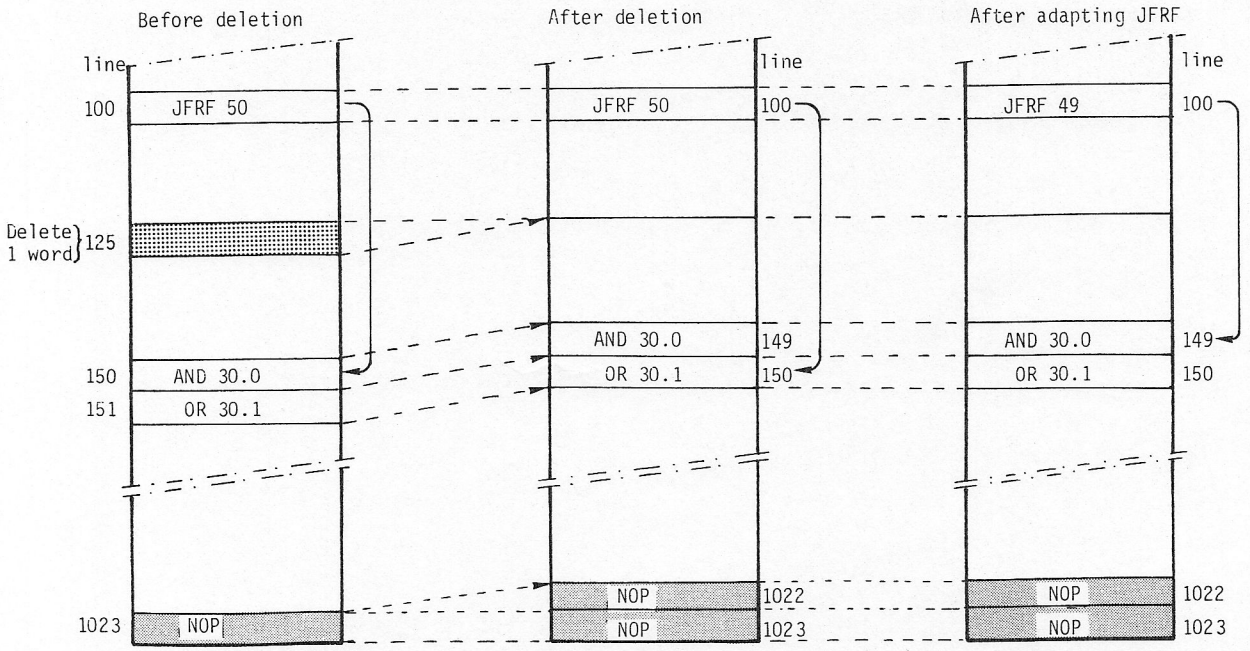


Fig. 4-8 Adaptation of JFRF (one program word deleted).

4.3.4 Inserting program words

Program words are inserted in an existing program using the INS(ERT) key. Insertion is done according to the steps described below.

1. Enter line number where the program word is to be inserted in the program. The 'RUN' LED will light.
2. Depress INS key. The abbreviation 'INS' will now appear in the DATA field.
3. Type in program word (instruction part and address part). Check word displayed in the INST field and SMA field.
4. Press ENTER key. Now the 'BUSY' LED will light. Meanwhile, all succeeding program words are shifted to the next higher line number to make place for the program word to be inserted until a NOP word is met, which will disappear. Insertion completed, the 'RUN' and 'BUSY' LED will both extinguish while the word 'EOJ' appears in the DATA field; also, the inserted word will be displayed.

To insert further program words following the word already inserted in the program, step 1 must be read as: "Depress S^+ key". Fig. 4-9 explains what happens when inserting one program word. Because a new word is entered on line no. 100, the program part, lines 100 to 1021, will shift to occupy lines 101 to 1022 (NOP word on line 1022 disappearing).

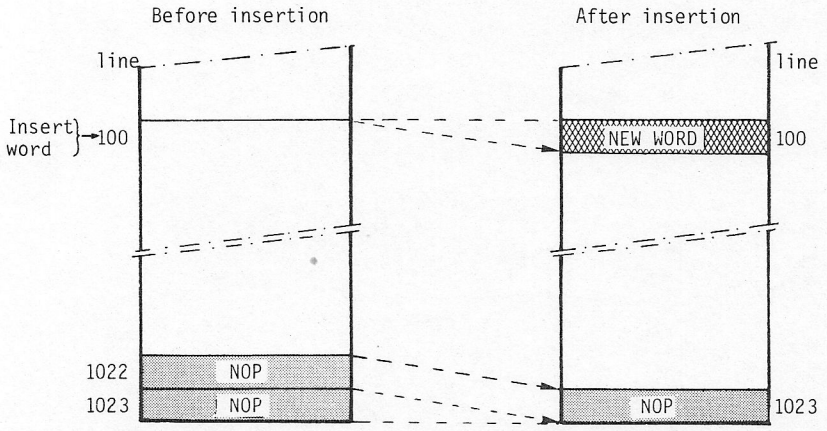


Fig. 4-9 Insertion of a program word for NOP words placed at the end of program.

Obviously, the number of NOP instructions available in the program must be at least equal to the number of words to be inserted, or the entire program will be confused. ('BUSY' LED remains on; press STOP key in this case.) Therefore, the golden rule of the previous section must be changed to:

It is wise practice to program hundred NOP words on the highest line numbers to have an ample margin.

Section 4.3.7 explains how this is done.

It should be noted that, with a large program memory and NOP words placed at the highest line numbers, insertion will take a longer time than in the case of a 1k memory.

In the following example an adaptation of the program given in Section 4.3.1 is worked out.

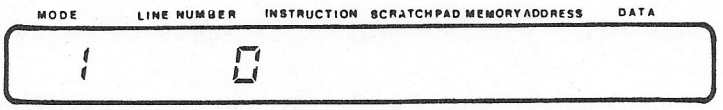
	line	program word		
INS	0000	AND 30.0		If 30.0 becomes HIGH
INS	0001	AND 30.1		AND 30.1 becomes HIGH
	0002	AND 1.1		AND 1 s clock pulse occurs
	0003	TRIG 39.0	]	make one count (units)
	0004	CNTU 12	]	
	0005	CNTU 13		When carry occurs make one count (tens).
	0006	STRB 16.0		Store state bit of tens counter on SMA 16.0.
	.	.		.
	.	.		.
	.	.		.

It is seen that two consecutive program words are to be inserted on line no. 0000.

Example 4-4 Inserting program words.

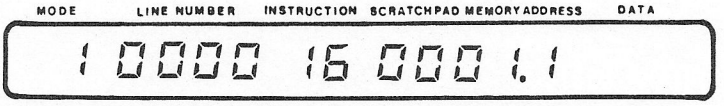
Choose line no. 0000:

[LINE] [0]



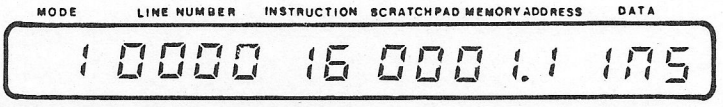
Check original program word

[ENTER]



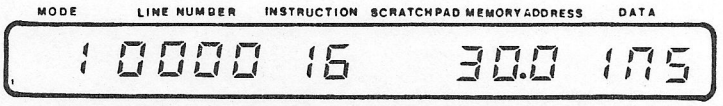
Depress INS key:

[INS]



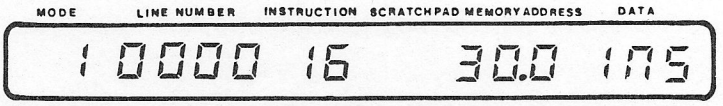
Type in new program word and check display:

[AND] [3] [0] [.] [0]

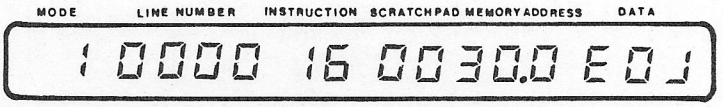


Start insert operation:

[ENTER]

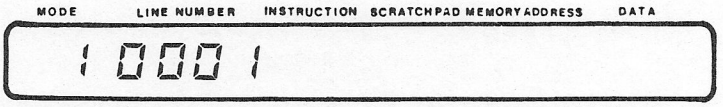


Insertion ready →



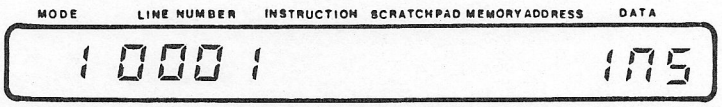
Take next line number:

[S+]



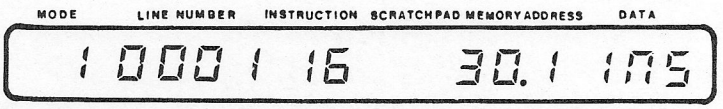
Depress INS key:

[INS]



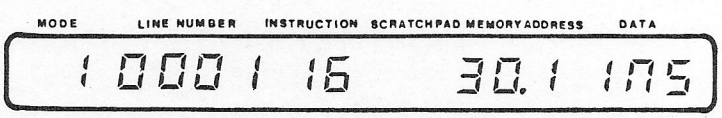
Type in next program word and check display:

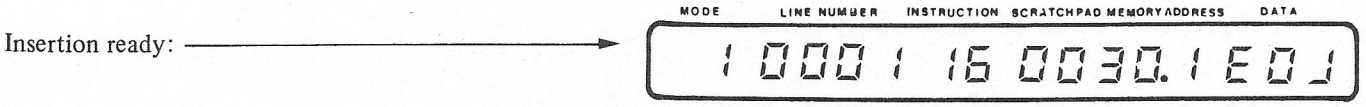
[AND] [3] [0] [.] [1]



Start insert operation:

[ENTER]





Modification of JUMP instructions. Since program words are shifted during insertion, it may be required to adapt JUMP instructions. As an example it is assumed that the program contains a jump from line 200 to 250 and that a new word is to be inserted on line 100. The jump must now be from line 201 to 251. Since the same number of lines must still be skipped, a relative jump need not be changed in this example. However, an absolute jump must be modified as JSAT (or JSAF) 251. Jumps are found using the search instruction, Section 4.3.6.

4.3.5 Entering macros

Macros are miniprograms that are generated on demand by the PU20 programming unit for storage in the RAM of the PC20. Benefits are: (1) shorter programming time, (2) easier programming. The user has only to define the scratchpad locations occurring in the macro. The PU20 contains six standard macros but up to ninety-three additional, "tailor-made" macros can be built in. Full details on macros are given in Chapter 6.

The six standard macros are specified in Table 4-3.

TABLE 4-3 The six standard macros.

macro number	description	number of lines
1	retriggerable three-decade delayed-on timer based on preset three-decade down counter	19
2	retriggerable three-decade delayed-off timer based on preset three-decade down counter	22
3	retriggerable three-decade one-shot based on preset three-decade down counter	20
4	multiplying program: 3 digits X 3 digits	33
5	dividing program: 6 digits (:) 3 digits	45
6	square-root extraction program for six-digit number	99

A macro is activated by depressing the MAC key, which causes symbol 'P' to appear in the INST field; then the relevant macro number is typed in on the digit keyboard. After the various scratchpad addresses have been entered using the digit keyboard, the abbreviation 'ASS' will be shown in the DATA field. Now, when the ASS(EMBLE) key is operated, the macro will be stored in the C-MOS RAM of the PC20 as part of the program. Upon completion of storage, the acronym 'EOJ' – for End Of Job – will appear in the DATA field as well as the last line of the macro program. On the next line number programming can now continue.

Macro 1: Retriggerable three-decade delayed-on timer.

To define the circuit, a set of questions is asked by the PU20, the questions numbers appearing in the DATA field. The user answers by typing in the appropriate scratchpad address on the digit keyboard then depressing the ENTER key for each question. One question having been answeerd, the next question number will appear until the circuit is fully defined.

The scratchpad addresses to be entered are:

1. Lowest of three consecutive four-bit scratchpad memory addresses (SMAs) of counter part, the lowest address containing the most significant digit (MSD); these addresses can be either *internal* or *output* addresses.
2. Four-bit *internal* SMA, which is used for input (location .1), delayed output (location .0) and for auxiliary signals (locations .2 and .3).
3. Lowest of three consecutive four-bit *input* addresses for three-digit preset value; also here, the MSD is placed on the lowest address.
4. Single-bit SMA for count input; this can be either the *internal* clock (see Table 4-4 below) or an *external* pulse source.

TABLE 4-4 SMAs of internal clock.

SMA	clock frequency
000.3	0,01 s
001.0	0,1 s
001.1	1 s
001.2	10 s
001.3	1 min

The following example will clarify what has been discussed above. Macro programming is assumed to start on line 7. The allocation of the scratchpad places is as shown in Fig. 4-10.

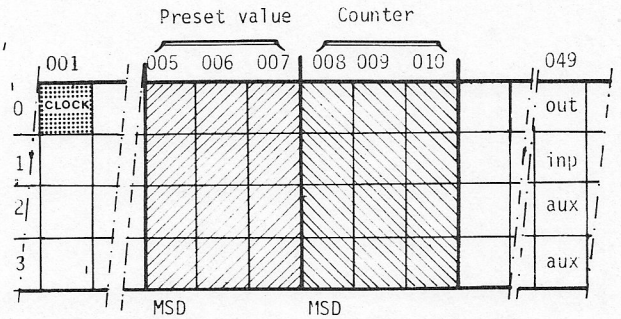
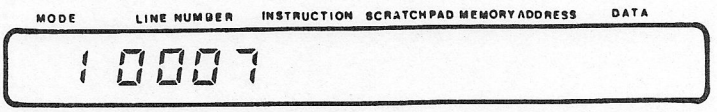


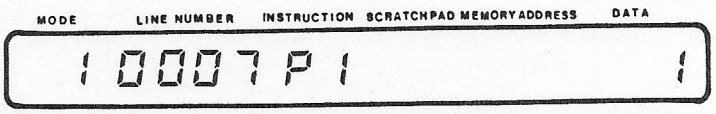
Fig. 4-10 Allocation of scratchpad addresses for delayed-on timer (Macro 1).

Example 4-5 Programming of Macro 1: Delayed-on timer.

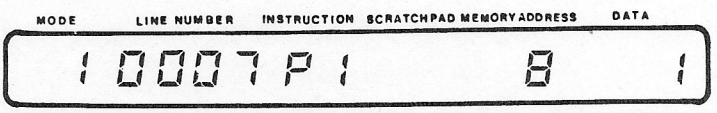
Next line no. selected
(previous program part ending on line 6).



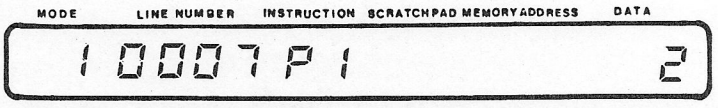
Activate Macro 1:
[MAC] [1] [ENTER]



QUE.1: WHERE TO PROGRAM COUNTER?
Answ.: Select SMAs 8 (MSD), 9 & 10:
[8]



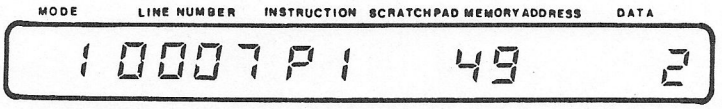
Enter answer:
[ENTER]



QUE.2: WHERE TO PROGRAM INPUT AND OUTPUT?

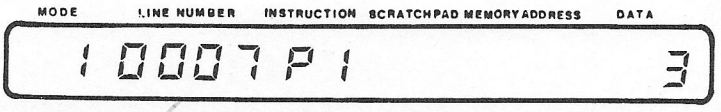
Answ.: Select SMA 49:

[4] [9]



Enter answer:

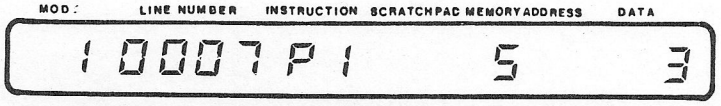
[ENTER]



QUE 3: WHERE TO PROGRAM PRESET-VALUE INPUTS?

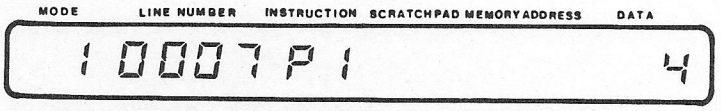
Answ.: Select SMAs 5 (MSD), 6 & 7:

[5]



Enter answer:

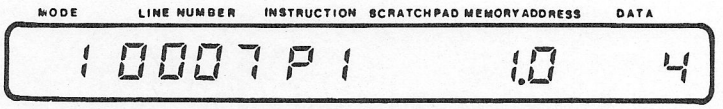
[ENTER]



QUE. 4: COUNT INPUT?

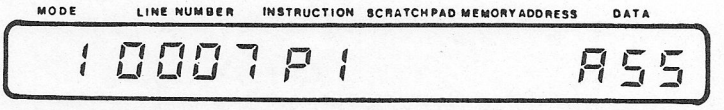
Answer: Select 0,1 s clock (Table 4-4):

[1] [.] [0]



Enter answer:

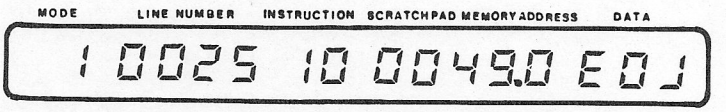
[ENTER]



Write macro into RAM:

[ASS]

Write-in completed ——————————►
(last macro program word shown)



Now continue programming on line 26:

[S+]

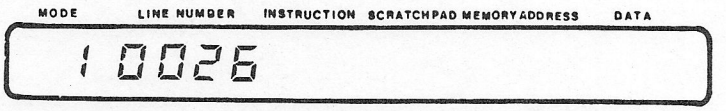


Fig. 4-11 illustrates the assembled software circuit and its characteristic waveforms. The program of this example is given below; its operation is explained in Chapter 6.

PROGRAM OF MACRO 1: DELAYED-ON TIMER.

```
#0007 17  049.1 [ANDNT]
#0008 08  049.0 [SETO ]
#0009 16  049.1 [ AND ]
#0010 01  049.2 [TRIG ]
#0011 30 0007   [JREF ]
#0012 13 0007   [FTCHD]
#0013 13 0006   [FTCHD]
#0014 13 0005   [FTCHD] (MSD)
#0015 14 0010   [STRD ]
#0016 14 0009   [STRD ]
#0017 14 0008   [STRD ]
#0018 16  049.1 [ AND ]
#0019 16  001.0 [ AND ]
#0020 17  049.0 [ANDNT]
#0021 01  049.3 [TRIG ]
#0022 06 0010   [CNTD ]
#0023 06 0009   [CNTD ]
#0024 06 0008   [CNTD ] (MSD)
#0025 10  049.0 [STRB ]
```

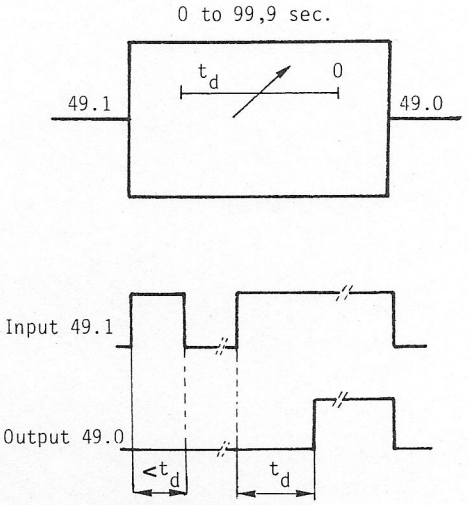


Fig. 4-11 Macro 1: Retriggerable software delayed-on timer and waveforms for Example 4-5.

Macro 2: Retriggerable three-decade delayed-off timer.

The functions of the scratchpad addresses to be typed in here are as described for Macro 1. See example 4-6 given for macro programming assumed to start on line 39. Fig. 4-12 shows the assigned scratchpad addresses.

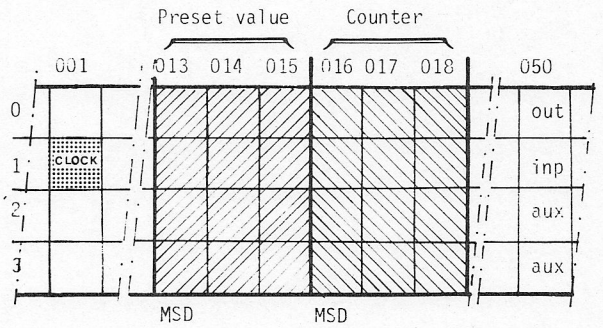
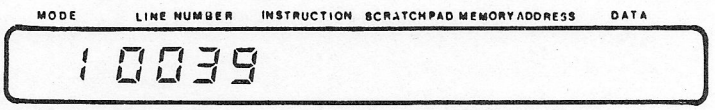


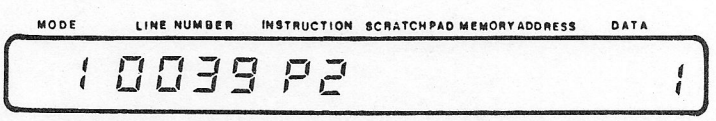
Fig. 4-12 Allocation of scratchpad addresses for delayed-off timer (Macro 2).

Example 4-6 Programming of Macro 2: Delayed-off timer.

Next line no. selected
(previous program part ending on line 38).



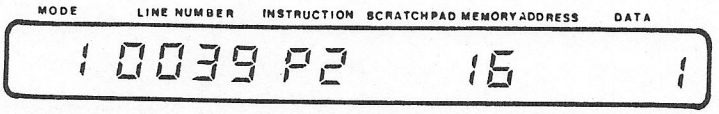
Activate Macro 2:
MAC 2 ENTER



QUE. 1: WHERE TO PROGRAM COUNTER?

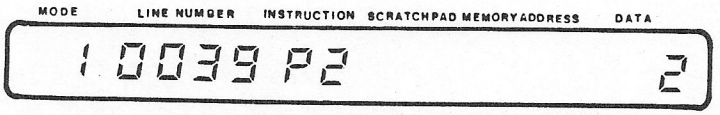
Answ.: Select SMAs 16 (MSD), 17 & 18:

[1] [6]



Enter answer:

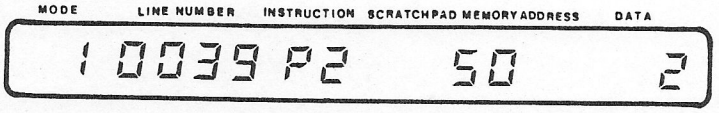
[ENTER]



QUE. 2: WHERE TO PROGRAM INPUT AND OUTPUT?

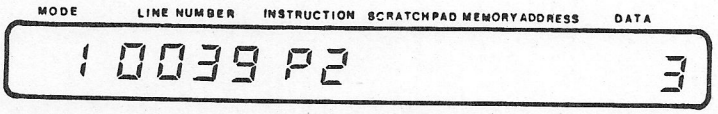
Answ.: Select SMA 50:

[5] [0]



Enter answer:

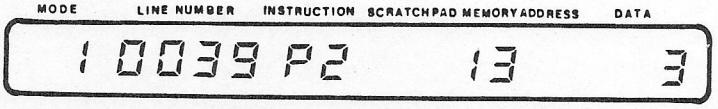
[ENTER]



QUE. 3: WHERE TO PROGRAM PRESET-VALUE INPUTS?

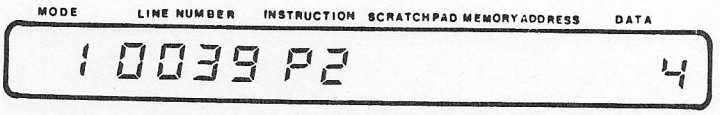
Answ.: Select SMAs 13 (MSD), 14 & 15:

[1] [3]



Enter answer:

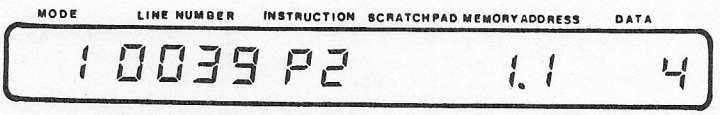
[ENTER]



QUE. 4: COUNT INPUT?

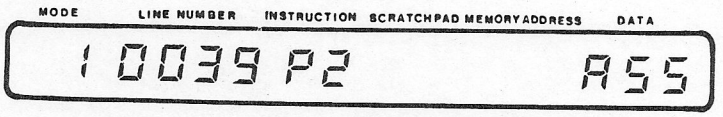
Answer: Select 1 s clock (table 4-4):

[1] [.] [1]



Enter answer:

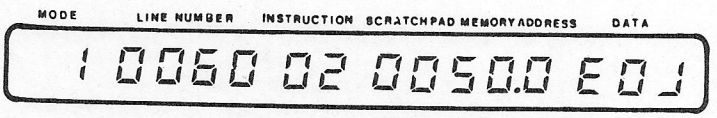
[ENTER]



Write macro into RAM:

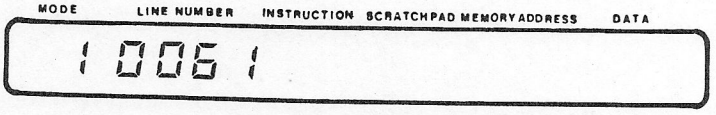
[ASS]

Write-in completed ——→
(last macro program word shown)



Now continue programming on line 61:

[S+]



The assembled software circuit is as shown in Fig. 4-13 together with its waveforms. The program of this example is as shown below; see Chapter 6 for a full discussion.

PROGRAM OF MACRO 2: DELAYED-OFF TIMER.

```
#0039 16  050.1 [ AND ]
#0040 09  050.0 [SET1 ]
#0041 17  050.1 [ANDNT]
#0042 01  050.2 [TRIG ]
#0043 30 0007   [JUMP ]
#0044 13 0015   [RTCHD]
#0045 13 0014   [RTCHD]
#0046 13 0013   [RTCHD]  (MSD)
#0047 14 0018   [STRD ]
#0048 14 0017   [STRD ]
#0049 14 0016   [STRD ]
#0050 17  050.1 [ANDNT]
#0051 16  001.1 [ AND ]
#0052 16  050.0 [ AND ]
#0053 01  050.3 [TRIG ]
#0054 06 0018   [CNTD ]
#0055 06 0017   [CNTD ]
#0056 06 0016   [CNTD ]  (MSD)
#0057 10  050.0 [STOP ]
#0058 30 0003   [JUMP ]
#0059 17  050.0 [ANDNT]
#0060 02  050.0 [ POL ]
```

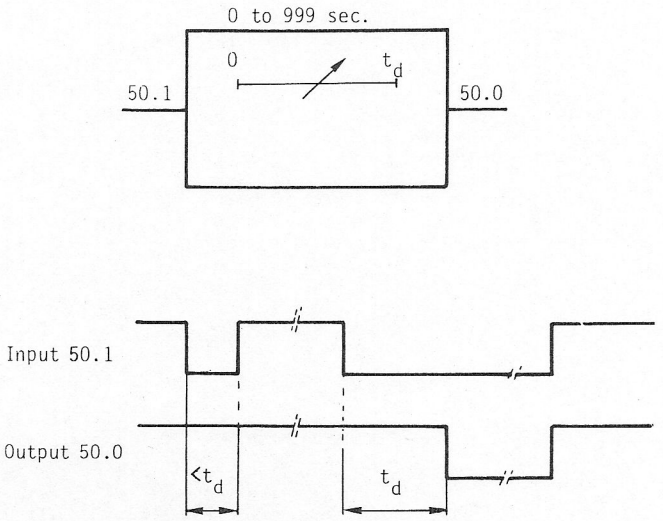


Fig. 4-13 Macro 2: Retriggerable software delayed-off timer and waveforms for Example 4-6.

Macro 3: Retriggerable three-decade one-shot.

The functions of the scratchpad addresses are as for Macro 1. In the example below macro programming starts at line 74. See Fig. 4-14 for allocation of scratchpad places.

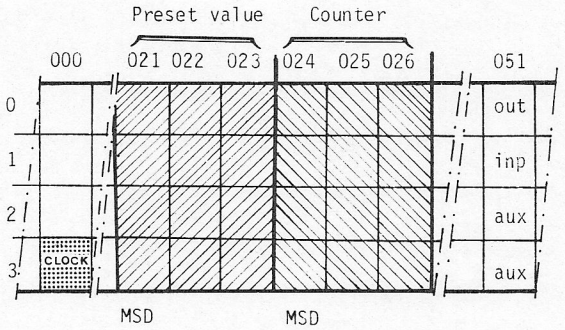
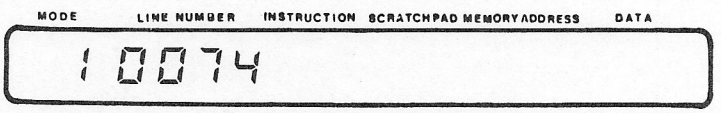


Fig. 4-14 Allocation of scratchpad addresses for one-shot (Macro 3).

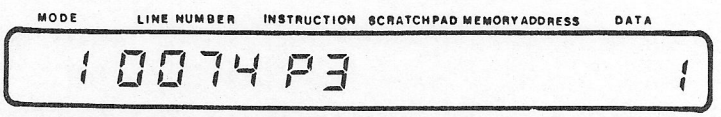
Example 4-7 Programming of Macro 3: One-shot.

Next line no. selected
(previous program part ending on line 73)



Activate Macro 3:

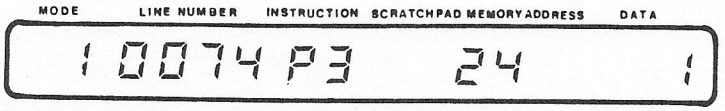
MAC 3 ENTER



QUE. 1: WHERE TO PROGRAM COUNTER?

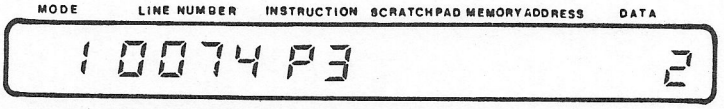
Answ.: Select SMAs 24 (MSD), 25 & 26:

[2] [4]



Enter answer:

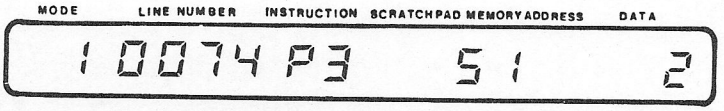
[ENTER]



QUE. 2: WHERE TO PROGRAM INPUTS AND OUTPUTS?

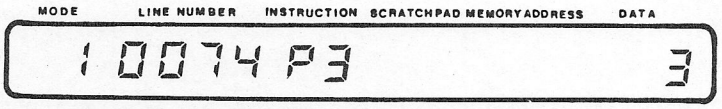
Answ.: Select SMA 51:

[5] [1]



Enter answer:

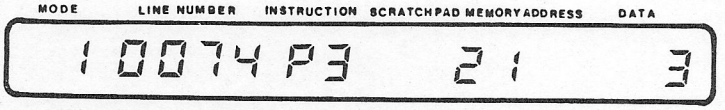
[ENTER]



QUE. 3: WHERE TO PROGRAM PRESET-VALUE INPUTS?

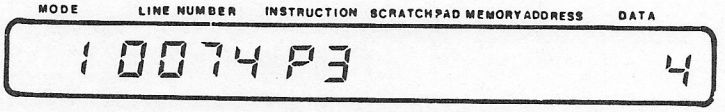
Answ,: Select SMAs 21 (MSD), 22 & 23:

[2] [1]



Enter answer:

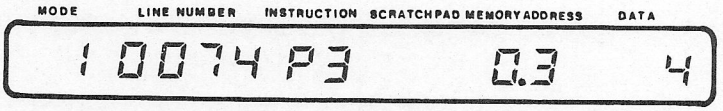
[ENTER]



QUE. 4: COUNT INPUT?

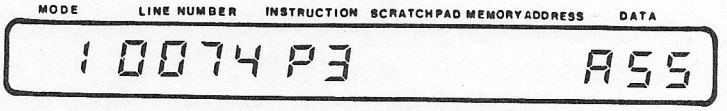
Answ.: Select 0,01 s clock (Table 4-4):

[0] [.] [3]



Enter answer:

[ENTER]

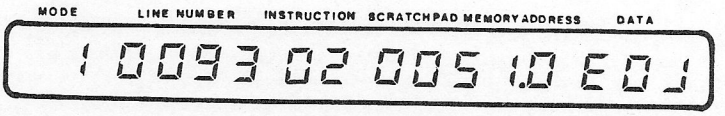


Write macro into RAM:

[ASS]

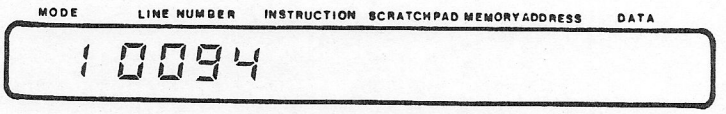
Write-in completed ——————→

(last macro program word shown)



Now continue programming on line 94:

[S+]



See Fig. 4-15 for the assembled software circuit and its waveforms. The program of this example follows here; see Chapter 6 for a full discusson.

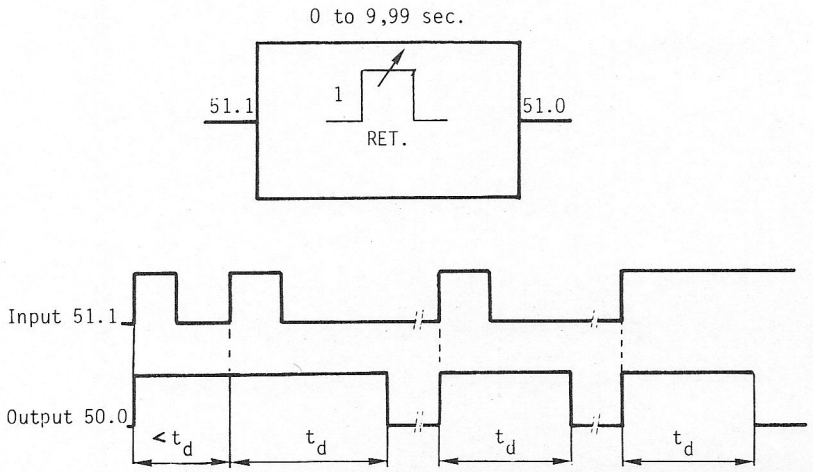


Fig. 4-15 Macro 3: Retriggerable software one-shot and waveforms for Example 4-7.

PROGRAM OF MACRO 3: ONE-SHOT.

```
#0074 16  051.1 [ AND ]
#0075 01  051.3 [TRIG ]
#0076 30 0008   [JFRF ]
#0077 13 0023   [FTCHD]
#0078 13 0022   [FTCHD]
#0079 13 0021   [FTCHD] (MSD)
#0080 14 0026   [STRD ]
#0081 14 0025   [STRD ]
#0082 14 0024   [STRD ]
#0083 02  051.0 [ EQL ]
#0084 16  051.0 [ AND ]
#0085 16  000.3 [ AND ]
#0086 01  051.2 [TRIG ]
#0087 06 0026   [CNTD ]
#0088 06 0025   [CNTD ]
#0089 06 0024   [CNTD ] (MSD)
#0090 10  051.0 [STRB ]
#0091 30 0003   [JFRF ]
#0092 17  051.0 [ANDNT]
#0093 02  051.0 [ EQL ]
```

Macro 4: Multiplying program.

This program can be expressed as:

A x B = C.

where: A = multiplicand (three digits).

B = multiplier (three digits),

C = product (max. six digits).

For this program the following scratchpad addresses should be defined:

1. Lowest (MSD) of three consecutive four-bit SMAs for three-digit multiplicand.
2. Lowest (MSD) of three consecutive four-bit SMAs for three-digit multiplier.
3. Lowest (MSD) of six consecutive four-bit SMAs to store the result (product).

As the macro program contains execute instructions solely, one or more logic instructions must precede to allow the program to run. This is worked out in the following example for macro programming starting on line 106. For allocation of scratchpad addresses see Fig. 4-16.

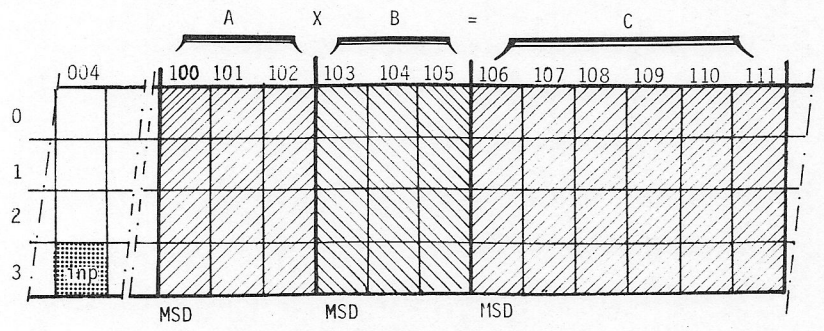
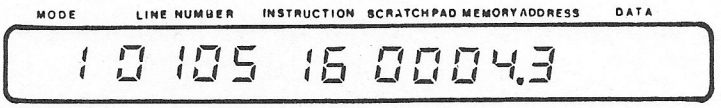


Fig. 4-16 Allocation of scratchpad addresses for multiplying program (Macro 4).

Example 4-8 Programming Macro 4: Multiplication.

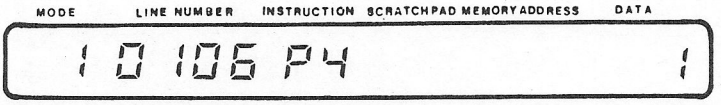
Write logic input condition on line 105
(previous program part ending on line 104):

[AND] [4] [.] [3] [ENTER]



Take next line no. and activate Macro 4:

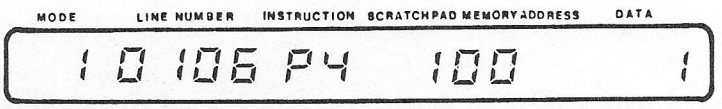
[S+] [MAC] [4] [ENTER]



QUE. 1: WHERE TO PROGRAM MULTIPLICAND PLACES?

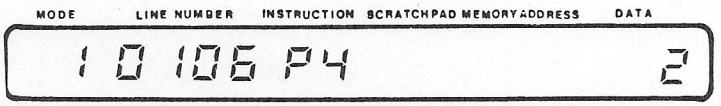
Answ.: Select SMAs 100 (MSD), 101 & 102:

[1] [0] [0]



Enter answer:

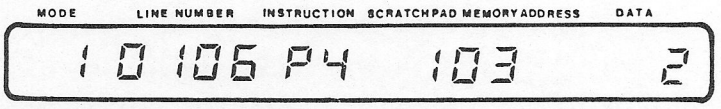
[ENTER]



QUE. 2: WHERE TO PROGRAM MULTIPLIER PLACES?

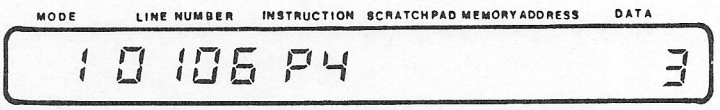
Answ.: Select SMAs 103 (MSD), 104 & 105.

[1] [0] [3]



Enter answer:

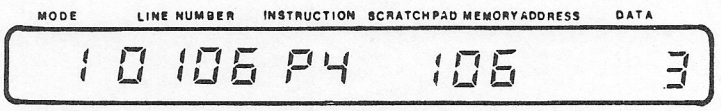
[ENTER]



QUE.3: WHERE TO STORE RESULT?

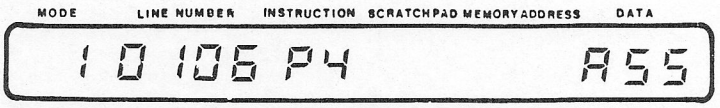
Answ.: Select SMAs 106 (MSD) to 111:

[1] [0] [6]



Enter answer:

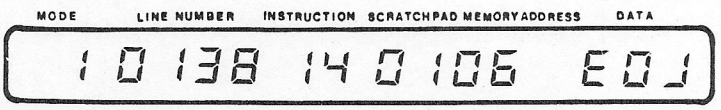
[ENTER]



Now write macro into RAM:

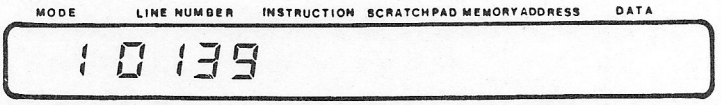
[ASS]

Write-in completed ——————→
(last macro program word shown)



Now continue programming on line 139:

[S+]



The multiplying program, which is explained in Chapter 6, is as given here (macro program starting on line 106).

PROGRAM OF MACRO 4: MULTIPLICATION.

```
#0105 16  004.3 [ AND ]  Input condition for macro
#0106 30 0033   [JREF ]
#0107 13 0102   [FTCHD]                 macro
#0108 13 0101   [FTCHD]
#0109 13 0100   [FTCHD](MSD) see Fig. 4-16
#0110 22 0105   [MULT ]
#0111 14 0111   [STRD ]
#0112 14 0110   [STRD ]
#0113 14 0109   [STRD ]
#0114 14 0108   [STRD ]
#0115 13 0102   [FTCHD]
#0116 13 0101   [FTCHD]
#0117 13 0100   [FTCHD](MSD)
#0118 22 0104   [MULT ]
#0119 10  000.1 [STRF ]
#0120 20 0110   [ ADD ]
#0121 20 0109   [ ADD ]
#0122 20 0108   [ ADD ]
#0123 14 0110   [STRD ]
#0124 14 0109   [STRD ]
#0125 14 0108   [STRD ]
#0126 14 0107   [STRD ]
#0127 13 0102   [FTCHD]
#0128 13 0101   [FTCHD]
#0129 13 0100   [FTCHD](MSD)
#0130 22 0103   [MULT ](MSD)
#0131 10  000.1 [STRF ]
#0132 20 0109   [ ADD ]
#0133 20 0108   [ ADD ]
#0134 20 0107   [ ADD ]
#0135 14 0109   [STRD ]
#0136 14 0108   [STRD ]
#0137 14 0107   [STRD ]
#0138 14 0106   [STRD ](MSD)
```

Macro 5: Dividing program.

The following formula is valid here:

$$A/B = Q + R/B,$$

where: A = dividend (six digits),

B = divisor (three digits),

Q = quotient (max. six digits),

R = remainder (max. three digits).

For this program the following scratchpad addresses must be entered:

1. Lowest (MSD) of six consecutive four-bit SMAs for six-digit dividend.
2. Lowest (MSD) of three consecutive four-bit SMAs for three-digit divisor.
3. Lowest (MSD) of six consecutive four-bit SMAs to store the result (quotient).

Since during execution of this program the dividend scratchpad locations are used as auxiliary places, the dividend will be destroyed and substituted by the remainder. Therefore, the program should be executed once only and so it is activated by a TRIG instruction as shown in the following example. Fig. 4-17 shows the scratchpad addresses used.

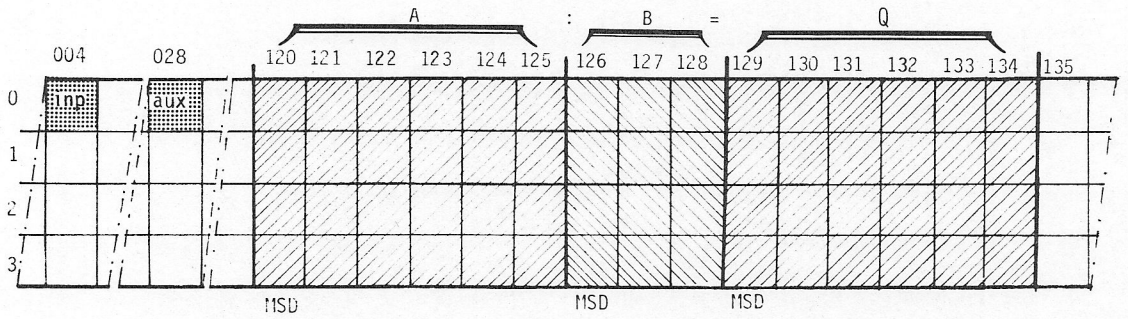
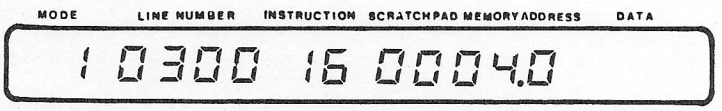


Fig. 4-17 Allocation of scratchpad addresses for dividing program (Macro 5).

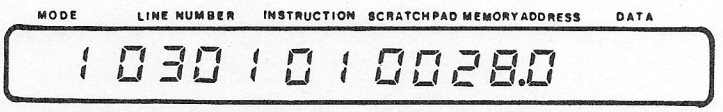
Example 4-9 Programming of Macro 5: Division

Write start condition on lines 300 & 301
(previous program part ending on line 299):

AND 4 . 0 ENTER

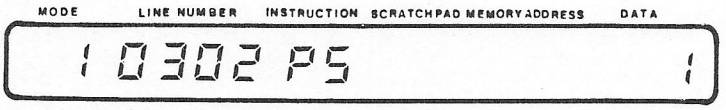


S+ TRIG 2 8 . 0 ENTER



Take next line no. and activate Macro 5:

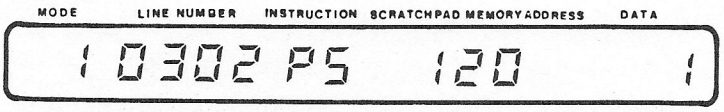
[S+] [MAC] [5] [ENTER]



QUE. 1: WHERE TO PROGRAM DIVIDEND PLACES?

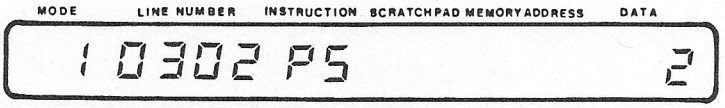
Answ.: Select SMAs 120 (MSD) to 125:

[1] [2] [0]



Enter answer:

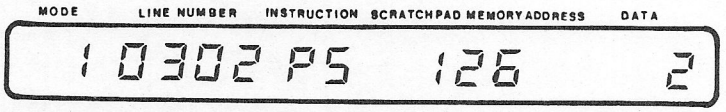
[ENTER]



QUE. 2: WHERE TO PROGRAM DIVISOR PLACES?

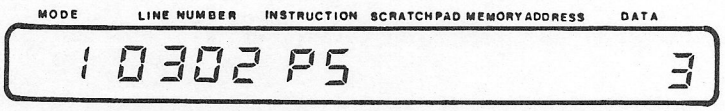
Answ.: Select SMAs 126 (MSD), 127 & 128:

[1] [2] [6]



Enter answer:

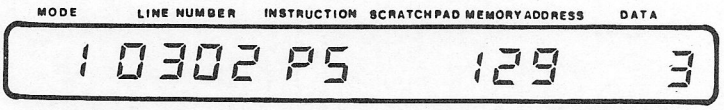
[ENTER]



QUE. 3: WHERE TO STORE QUOTIENT?

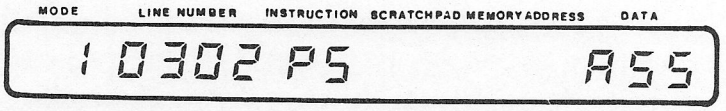
Answ.: Select SMAs 129 (MSD) to 134:

[1] [2] [9]



Enter answer:

[ENTER]

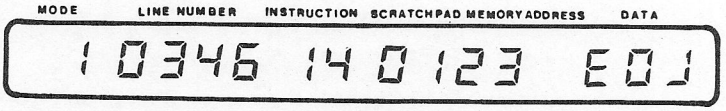


Write macro into RAM:

[ASS]

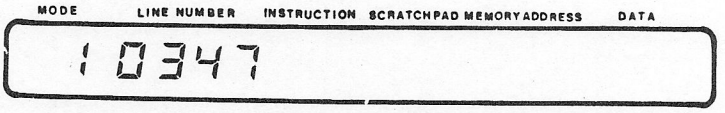
Write-in completed ——————→

(last macro program word shown)



Now continue programming on line 347:

[S+]



The dividing program is as given below (macro program starting on line 302). For a full discussion see Chapter 6.

PROGRAM OF MACRO 5: DIVISION.

```
#0300 16  004.0 [ AND ] Input condition for macro
#0301 01  028.0 [TRIG ]
#0302 30 0045   [JFRF ]          macro
#0303 13 0134   [FTCHD]
#0304 14 0134   [STRD ]
#0305 14 0134   [STRD ]
#0306 13 0123   [FTCHD]
#0307 13 0122   [FTCHD]
#0308 13 0121   [FTCHD]
#0309 13 0120   [FTCHD](MSD)
#0310 23 0128   [ DIV ]
#0311 23 0127   [ DIV ]
#0312 23 0126   [ DIV ](MSD)
#0313 14 0132   [STRD ]
#0314 14 0131   [STRD ]
#0315 14 0130   [STRD ]
#0316 14 0129   [STRD ](MSD)
#0317 17  000.0 [ANDNT]
#0318 20 0134   [ ADD ]
#0319 14 0123   [STRD ]
#0320 14 0122   [STRD ]
#0321 14 0121   [STRD ]
#0322 14 0120   [STRD ]
#0323 13 0124   [FTCHD]
#0324 13 0123   [FTCHD]
#0325 13 0122   [FTCHD]
#0326 13 0121   [FTCHD]
#0327 23 0128   [ DIV ]
#0328 23 0127   [ DIV ]
#0329 23 0126   [ DIV ](MSD)
#0330 14 0133   [STRD ]
#0331 20 0134   [ ADD ]
#0332 14 0124   [STRD ]
#0333 14 0123   [STRD ]
#0334 14 0122   [STRD ]
#0335 13 0125   [FTCHD]
#0336 13 0124   [FTCHD]
#0337 13 0123   [FTCHD]
#0338 13 0122   [FTCHD]
#0339 23 0128   [ DIV ]
#0340 23 0127   [ DIV ]
#0341 23 0126   [ DIV ](MSD)
#0342 14 0134   [STRD ]
#0343 20 0120   [ ADD ]
#0344 14 0125   [STRD ]
#0345 14 0124   [STRD ]
#0346 14 0123   [STRD ]
#0347 00        [ NOP ]
```

Macro 6: Square-root extraction program.

The square-root extraction program is carried out according to a converging series expressed as:

$$B_{n+1} = \tfrac{1}{2}(B_n + A/B_n).$$

where: B_n = nth term of series.

B_{n+1} = (n+1)th term of series.

A = six-digit number, the square root of which is to be found.

The square root will be displayed as a three-digit number preceded by its negative exponent: 0, 1 or 2. So, the square root of 10, which is 3,1623, will be shown as 2316 (316/100). The square root of 20000 is 141,42; this will appear as 0141.

Here, the following scratchpad addresses must be specified:

1. Lowest (MSD) of six consecutive four-bit SMAs for six-digit number.
2. Lowest of five consecutive four-bit SMAs to display exponent and three-digit square root (MSD on lowest address), the fifth SMA being used as an auxiliary address.
3. Lowest of six consecutive four-bit SMAs to store, amongst others, intermediate results.

As in the previous case, the program must be activated by a TRIG instruction as illustrated in the example (allocation of scratchpad addresses is shown in Fig. 4-18).

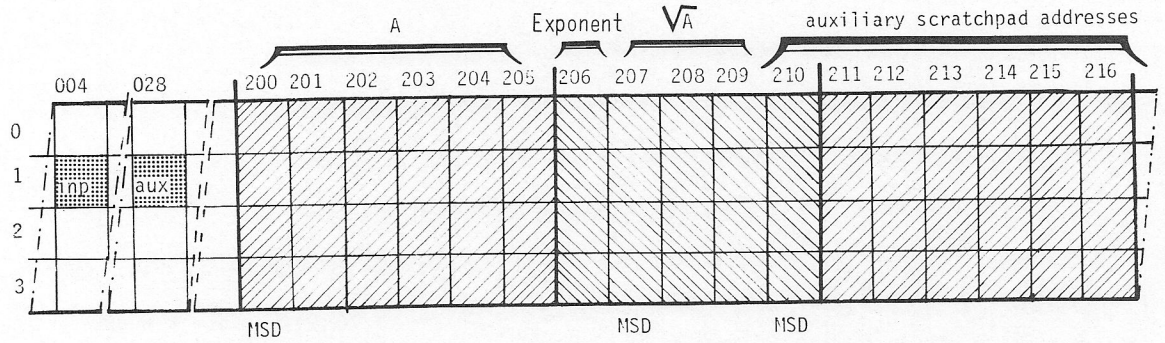
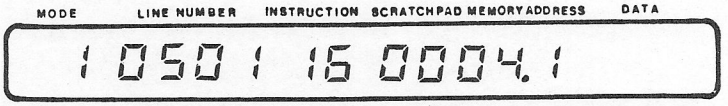


Fig. 4-18 Allocation of scratchpad addresses for square-root extraction program (Macro 6).

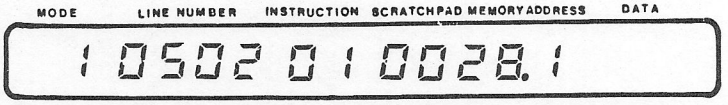
Example 4-10 Programming of Macro 6: Extracting square root.

Write start condition on lines 501 & 502
(previous program part ending on line 500):

[AND] [4] [.] [1] [ENTER]

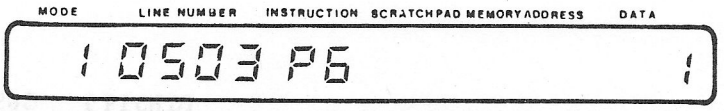


[S+] [TRIG] [2] [8] [.] [1] [ENTER]



Take next line no. and activate Macro 6:

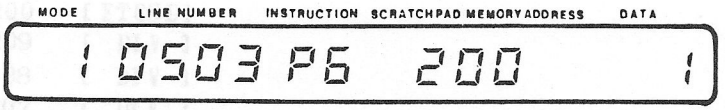
[S+] [MAC] [6] [ENTER]



QUE. 1: WHERE TO PROGRAM PLACES FOR SIX-DIGIT NUMBER?

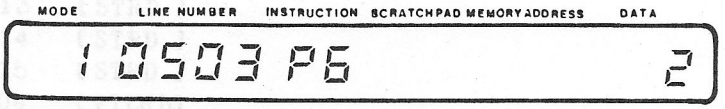
Answ.: Select SMAs 200 (MSD) to 205:

[2] [0] [0]



Enter answer:

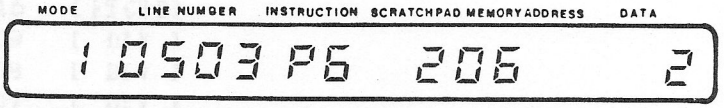
[ENTER]



QUE. 2: WHERE TO STORE RESULT?

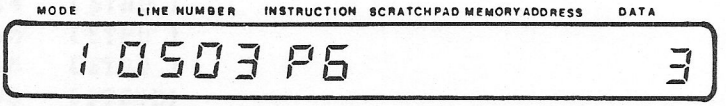
Answ.: Select SMAs 206 (exponent), 207 (MSD), 208 and 209, additionally aux. address 210:

[2] [0] [6]



Enter answer:

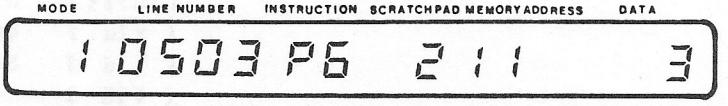
[ENTER]



QUE. 3: WHERE TO PROGRAM SUNDRY AUX. PLACES?

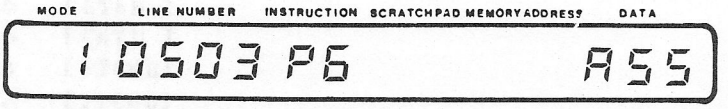
Answ.: Select SMAs 211 to 216:

[2] [1] [1]



Enter answer:

[ENTER]

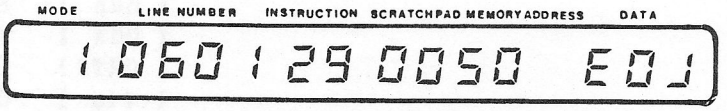


Write macro into RAM:

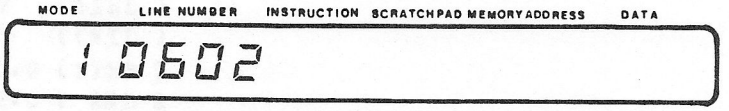
[ASS]

Write-in completed ——————→

(last macro program word shown)



Now continue programming on line 602:



The square-root extraction program, which is discussed in Chapter 6, follows here (macro program starting on line 503).

4.3.6 Search instruction

As seen before, JUMP instructions must, in most cases, be modified after inserting or deleting program words. JUMP instructions are easily found using the "Search instruction" facility of the PU20. Another useful application of this facility is to find the LSTIO instruction if we want to extend the program. "Search instruction" is initiated by operating the VFY (verify) key. The following steps are taken:

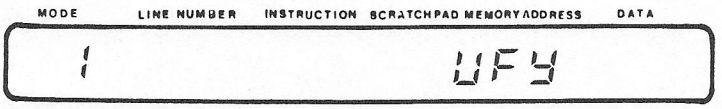
1. Depress VFY key. The 'RUN' LED will light and the abbreviation 'VFY' will appear in the SMA field.
2. Depress LINE key.
3. Type in on the digit keyboard the line number at which "Search instruction" must start (line number appearing in LINE field), then operate ENTER key.
4. Depress DATA key.
5. Type in on the digit keyboard the length of program in which the specific instruction is to be found starting at line number chosen in step 3. That is, depress key 1, 2, etc. for 1k, 2k, etc. length of program (up to 8k), each block of 1k containing 1024 line numbers. The typed-in digit will appear in the DATA field. Then press ENTER key.
6. Depress corresponding instruction key. Instruction number is displayed in INST field.
7. Depress START key. The LINE and SMA fields are blanked and 'VFY' will now appear in the DATA field. The 'BUSY' LED will be lit until the first line number containing the relevant instruction is found. That line number and its contents will be shown on the display.
8. Depress S^+ key to find next line number where the instruction appears. The above procedure will now be repeated. If in the chosen program part a further coincidence is not found, the 'RUN' LED will extinguish together with the 'BUSY' LED and 'EOJ' will be displayed in the DATA field. Contents of line number will stay on the display.
9. If 'EOJ' is not displayed, depress S^+ key again to repeat procedure, etc.

An example clarifying the use of the "Search instruction" facility is given here. The program is assumed to start as shown in Section 4.3.1, that is, a JSAT instruction is programmed on line 6. It is further assumed that the program (2k length) contains one more JSAT instruction on line 1005.

Example 4-11 Finding JSAT instruction.

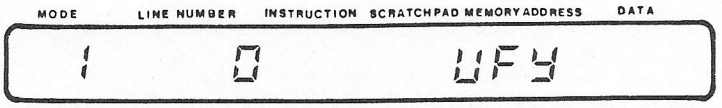
Press VFY key to start "Search instruction":

[VFY]



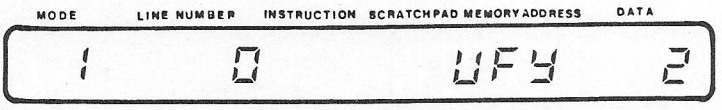
Select first line no.:

[LINE] [0] [ENTER]



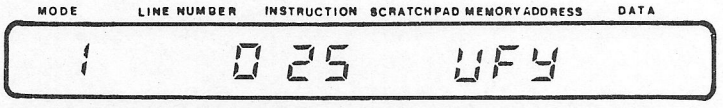
Choose program length
(entire program searched here):

[DATA] [2] [ENTER]



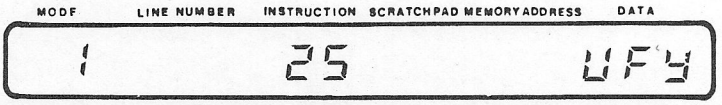
Depress instruction key:

[JSAT]

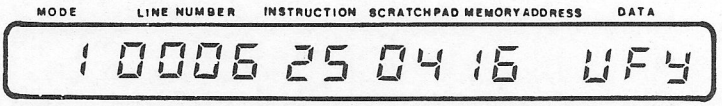


Depress START key:

[START]

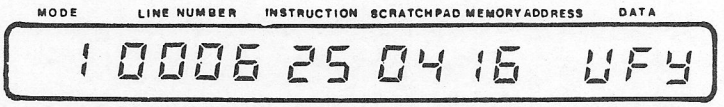


Line no 0006 found ——————→

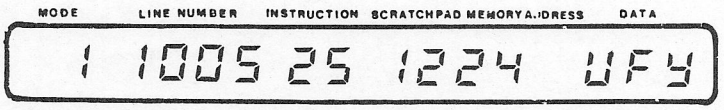


Depress S+ key:

[S+]

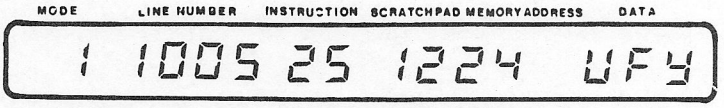


Next (last) line no. found ——————→

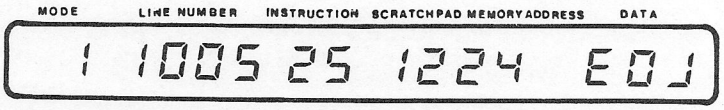


Depress S+ key:

[S+]



No more JSAT instructions found ——————→



4.3.7 How to start programming

Clearly, from previous sections, two major rules must be observed in programming the PC20:

- NOP words must be available in the program or deletion and insertion cannot be done (Sections 4.3.3 and 4.3.4).
- The program must be terminated with a LSTIO and END instruction; otherwise, the PC20 will never enter its I/O phase.

When the following steps are taken, problems will not be encountered:

1. Program a NOP word on last line.
2. Go back one hundred lines and issue 'DELETE 99' command by depressing the keys 'DEL', '9', '9' and 'ENTER' (cf. Section 4.3.3). Thus, one hundred NOP words will be generated at the end of the program. This will normally be adequate for insertion of program words.
3. After 'EOJ' has appeared in the DATA field, step one line number back (depressing S^- key) and program an END instruction.
4. Step back one line number and program a LSTIO instruction.
5. Program a few NOP words every 500 or 1000 lines. This is especially useful in the case of a large program memory because it will reduce the time required for deletion or insertion. These words will successively disappear as programming proceeds.
6. Start programming on line 0000. If a small program part is to be tested terminate that part temporarily with a LSTIO and END instruction.
7. Terminate with the LSTIO and END instruction upon completion of programming.

4.4 Monitoring 'continuous'

It is in the monitoring 'continuous' mode that the PC20 system comes into its own; here, it operates as a controller. In the previous sections we have seen how, in the edit mode, a program is entered into the C-MOS RAM of the PC20. Since the program terminates with a LSTIO and END instruction, I/O scanning and data processing will alternate here. That is, the system will work in its normal, running condition. The program can be tested on-line and, if performance is not satisfactory, changes can be made.

The monitoring 'continuous' mode offers the following facilities:

- Line monitoring (key switch can be off).
- SMA monitoring (key switch can be off).
- Three-digit SMA monitoring (key switch can be off).
- Changing program words (key switch on).
- Entering data into the scratchpad memory (key switch on).

With three-digit SMA monitoring, the contents of three adjacent scratchpad addresses are simultaneously displayed; this is useful to check, amongst others, the performance of timers, counters, etc. The facility of entering data into the scratchpad permits simulation of input signals or data sources; i.e. set-point values not yet available at the time of program testing can be entered.

For a proper interpretation of the DATA display, distinction must be made between the case of a single-bit and a four-bit scratchpad address. In the case of a one-bit instruction, a *single-bit* scratchpad location will be addressed and its contents, 0 or 1, will be shown in the *centre* of the DATA field. If a four-bit instruction is issued, the contents of the relevant *four-bit* scratchpad address will be shown in the *left-hand* part of the DATA field in hexadecimal notation; see Table 4-5. This is further clarified in the example. Fig. 4-19, for scratchpad address 004.

In Fig. 4-19a, one bit place 004.0 of scratchpad location 004 has been addressed; here, the state of only one DATA LED will be of interest. Because the contents of the bit place is '1', the corresponding DATA LED will be lit. In Fig. 4-19b, the entire contents of the four-bit location is displayed. This will now correspond to the state of all four DATA LEDs; see Table 4-6.

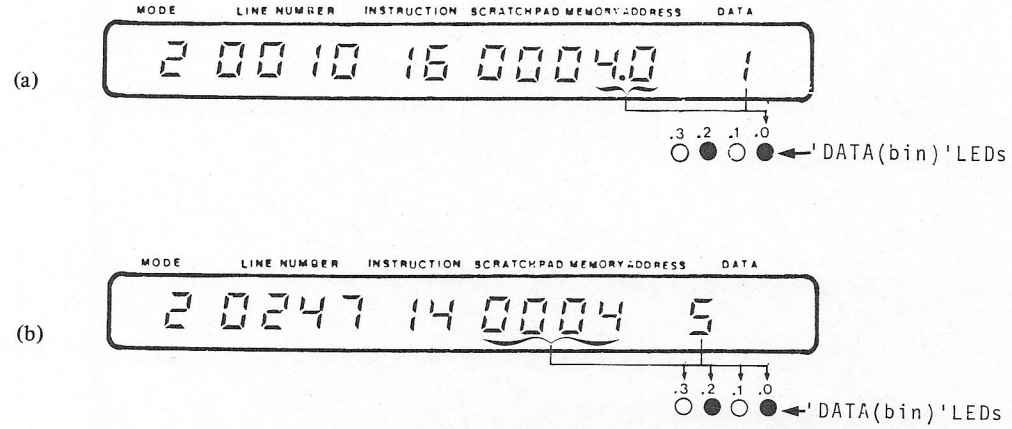


Fig. 4-19 Interpretation of DATA field display: (a) contents of one-bit scratchpad address displayed in centre of DATA field; (b) contents of four-bit scratchpad address displayed in left-hand part of DATA field.

TABLE 4-5 The hexadecimal display (left-hand part in DATA field).

number	hexadecimal display
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	.b
12	C
13	.d
14	E
15	F

TABLE 4-6 Weight of DATA LEDs for hexadecimal and BCD presentation.

DATA LED	weight
.0	1
.1	2
.2	4
.3	8

4.4.1 Line monitoring

For line monitoring the key switch can be in its OFF position. The line number to be monitored is selected by depressing the LINE key, then typing in that line number on the digit keyboard. Line searching commences when the ENTER key is depressed ('BUSY' LED on). As soon as a coincidence occurs, the program word – instruction and scratchpad address part – as well as the contents of that scratchpad address (Fig. 4-19) will be displayed. The 'BUSY' LED stays on indicating a continuous transfer of scratchpad data to the PU20. The display will be updated whenever a change in data of the addressed scratchpad location occurs. By depressing the S^+ or S^- key, the program word of the adjacent line number and the appertaining scratchpad data will be shown as described above.

If the selected line number is skipped (JUMP instruction active), a coincidence cannot occur and the program word and data of that line number will not be shown when the ENTER key is depressed. To create a coincidence, the UDC key must be used. Now only the program word will be displayed; the DATA field will stay blanked because the data in that scratchpad location do not contribute to the execution of the program. Before selecting a new line number, the STOP key must be depressed for the PU20 to come out of UDC.

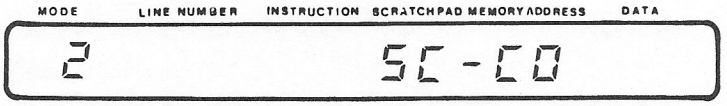
To illustrate what has been discussed above, the following program will be used as an example.

line	program word		
⋮	⋮		
0020	AND 4.1		If 4.1 is HIGH
0021	ANDNT4.2		and 4.2 is LOW
0022	EQL 8.1		8.1 will be HIGH.
0023	FTCHC 15	}	Store '15' on SMA 100.
0024	STRD 100	}	
0025	TRIG 101.1		If 101.1 goes HIGH
0026	JFRF 3		execute line nos. 0027 & 0028 (no jump).
0027	CNTD 102		When no jump occurs make one down-count and
0028	STRB 101.2		store state bit of counter 102 on SMA 101.2.
0029	ANDNT4.1		If 4.1 is HIGH, that is $\overline{4.1}$ LOW
0030	EQL 8.2		8.2 will be LOW.
⋮	⋮		

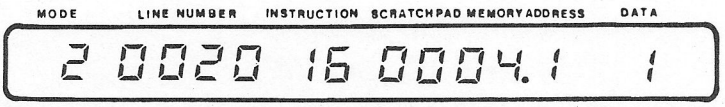
In the example worked out below SMAs (inputs) 4.1 and 4.2 are assumed to have states '1' and '0', respectively, and SMA 101.1 is assumed to have the '1' state.

Example 4-12 On-line display of program

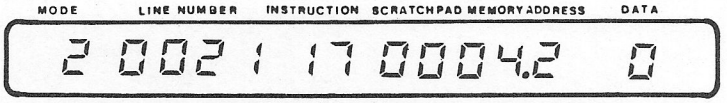
Monitoring 'continuous' mode selected
(see Section 4.2.2)



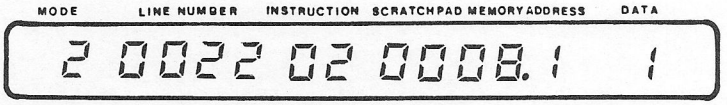
Select line no. 20:
[LINE] [2] [0] [ENTER]



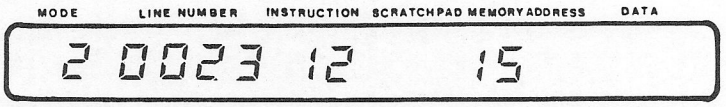
Take next line no.:
[S+] [ENTER]



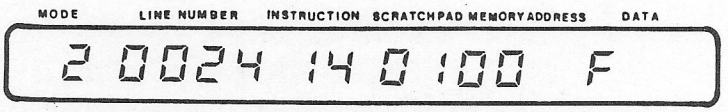
Take next line no.:
[S+] [ENTER]



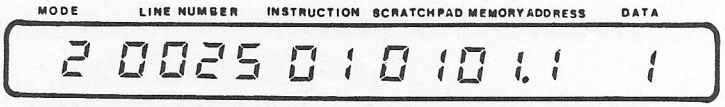
Take next line no.:
[S+] [ENTER]



Take next line no.:
[S+] [ENTER]

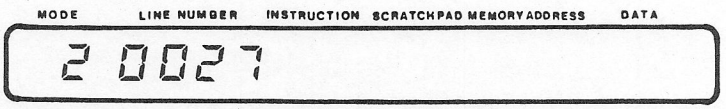


Take next line no.:
[S+] [ENTER]

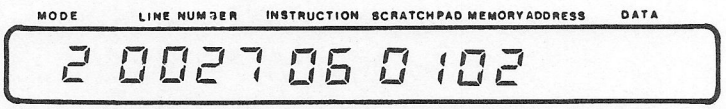


Take next line no.:
[S+] [ENTER]

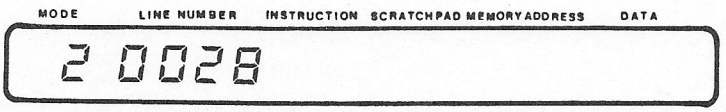
Take next line no.-
[S+] [ENTER]



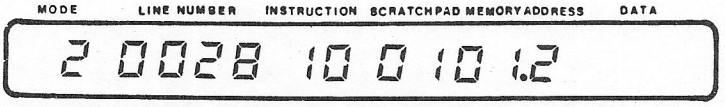
Because JUMP instruction is active here,
use UDC key:
[UDC]



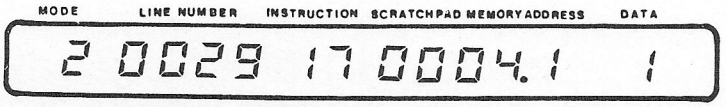
Take next line no.:
[STOP] [S+] [ENTER]



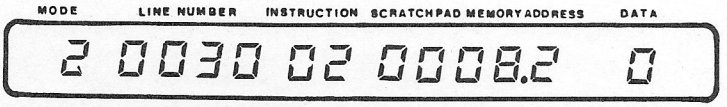
JUMP inst. active so use UDC key:
[UDC] [ENTER]



Take next line no.:
[STOP] [S+]
(When UDC key has not been depressed STOP key need not be operated)



Take next line no.:
[S+] [ENTER]



4.4.2 SMA monitoring

In the previous section, we have seen how to display the program word on any line, given the line number. Now we start with the scratchpad address and we want to know on which line number(s) this address appears. Because this is a search command, somewhat similar to the "Search instruction" command dealt with in Section 4.3.6, we must type in the line number at which searching is to commence. The scratchpad location is addressed by depressing the SMA key then typing in the scratchpad address on the digit keyboard. SMA searching will start when the ENTER key is operated. As soon as SMA coincidence occurs, the first line number where the typed-in scratchpad address appears will be displayed together with its program word and the scratchpad data. Also, the 'BUSY' LED will light and stay on to indicate that the DATA display will be updated whenever a change in scratchpad data occurs. Depressing the S^+ key, then the ENTER key will cause the next line number with the typed-in scratchpad location and its contents to be displayed. For SMA monitoring the key switch can be in its OFF position. Summarizing, the following steps are taken:

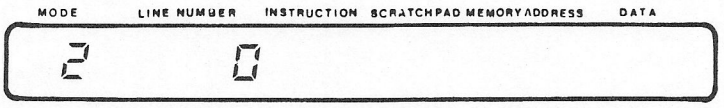
1. Depress LINE key, then type in start line number on the digit keyboard. This line number will appear in the LINE field.
2. Depress SMA key and type in scratchpad address on the digit keyboard. This address will be displayed in the SMA field.
3. Depress ENTER key to display first line number containing typed-in scratchpad location.
4. Depress S^+ key then ENTER key to display next line number with typed-in scratchpad location.
5. If INST field is not blanked press S^+ key then ENTER key to find next line number. If INST field is blanked no more line numbers containing the chosen scratchpad address will be present in the program.
6. SMA searching completed, depress STOP key to terminate "Search SMA" mode.

Taking the program part of Section 4.4.1, we will show in the example that follows how scratchpad location 4.1 is found.

Example 4-13 SMA monitoring: coincidence occurring.

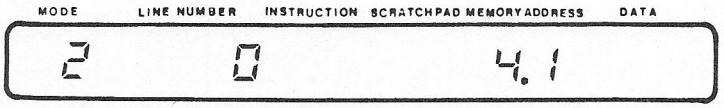
Type in start line no.:

[LINE] [0]



Type in scratchpad address:

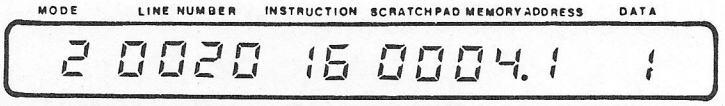
[SMA] [4] [.] [1]



Find 1st line no. containing SMA 4.1:

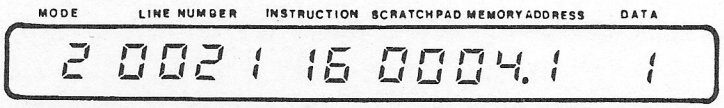
[ENTER]

(DATA display follows (4.1)*)



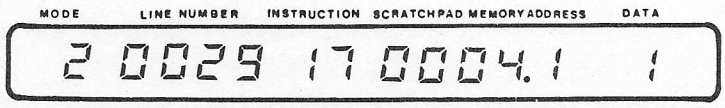
Find 2nd line no. containing SMA 4.1:

[S+] (display has no significance)



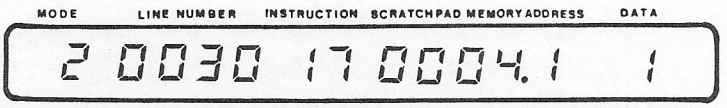
[ENTER]

(DATA display follows (4.1)*)

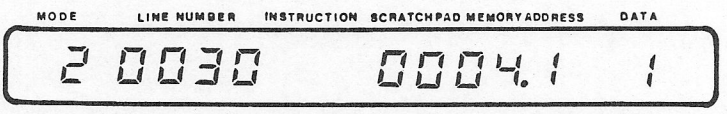


Find 3rd line no. containing SMA 4.1:

[S+] (display has no significance)



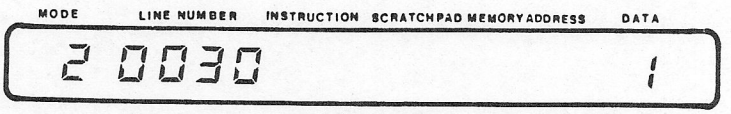
[ENTER]



Because INST field is blanked, other line nos. containing SMA 4.1 do not appear in the program.

Depress STOP key to end "Search SMA" mode:

[STOP]



As in the case of line monitoring, it may appear that the chosen scratchpad address cannot be found because it occurs in a skipped program (jump active). To generate a coincidence, the UDC key is again operated. As a result, the scratchpad data will be shown, but the INST field remains blanked here. Further, the start line number will stay in the display. The DATA display will be updated whenever the UDC key is depressed ('BUSY' LED staying on); this is useful to investigate, in the case of an input address, if the input is properly transferred to that address. Also here the STOP key must be depressed for the PU20 to come out of the UDC mode.

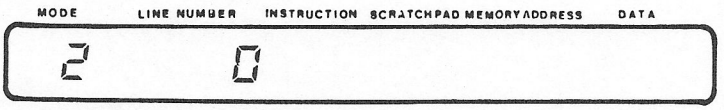
The following example, which is based on the program part of Section 4.4.1, will clarify the procedure. It is assumed that the contents of SMA 102, a down counter, are to be monitored.

* '(4.1)' means "contents of SMA 4.1".

Example 4-14 SMA monitoring; no coincidence

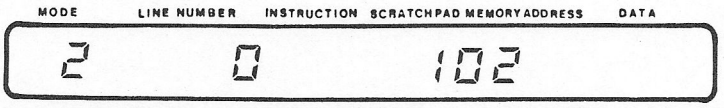
Type in start line no.-

[LINE] [0]



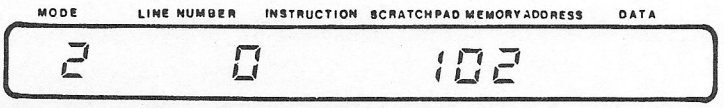
Type in scratchpad address:

[SMA] [1] [0] [2]



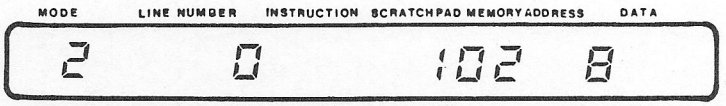
Find (102)*:

[ENTER]



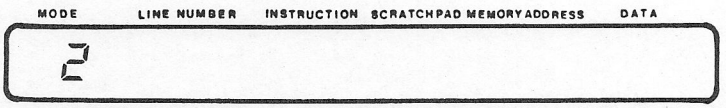
Now depress UDC key:

[UDC]



Press STOP key to terminate UDC mode:

[STOP]



* '(102)' means "contents of SMA 102".

4.4.3 Three-digit SMA monitoring

As an extension to SMA monitoring, the PU20 can be made to monitor three consecutive four-bit scratchpad addresses simultaneously, the result being displayed in the DATA field (key switch can be off). Three-digit SMA monitoring is useful to check the performance of timers, counters and one-shot software circuits. Programming of three-digit monitoring must occur as explained below.

Programming. Three adjoining scratchpad locations are taken, that with the *lowest* address carrying the *most significant* decade. For a three-decade counter programming could occur as follows:

```
line    program word
 :        :
 :        :
0500    AND   1.1          When 1 s clock pulse occurs
0501    TRIG  99.3    ┐
0502    CNTU  104 (LSD) ┘  make one count (units).
0503    CNTU  103          When carry occurs make one count (tens).
0504    CNTU  102 (MSD)    When carry occurs make one count (hundreds).
0505    STRB  115.1        Store counter state bit on SMA 115.1.
 :        :
 :        :
```

Monitoring. For three-digit SMA monitoring, the following steps must be observed:

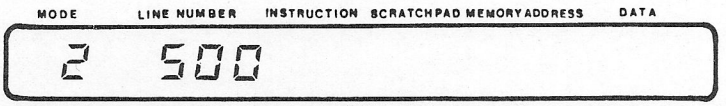
1. Depress LINE key and type in start line number on digit keyboard. This line number must be *lower* than those describing the three consecutive scratchpad locations to be monitored.
2. Depress SMA key, then MAC key and type in the *lowest* of the three consecutive scratchpad addresses.
3. Depress ENTER key. The counter contents will now be displayed in the DATA field as a three-digit number; 'BUSY' LED will light to indicate that the data are being updated.
4. Depress STOP key to terminate three-digit SMA monitoring.

Simultaneous monitoring of the three consecutive four-bit scratchpad locations (see program above) will now be elaborated in an example.

Example 4-15 Three-digit SMA monitoring.

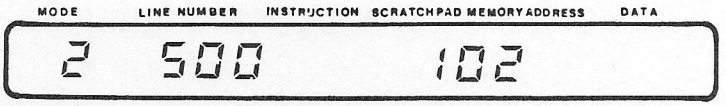
Select start line no.:

[LINE] [5] [0] [0]



Select lowest SMA:

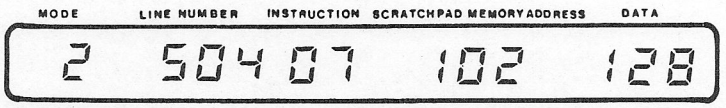
[SMA] [MAC] [1] [0] [2]



Display counter contents:

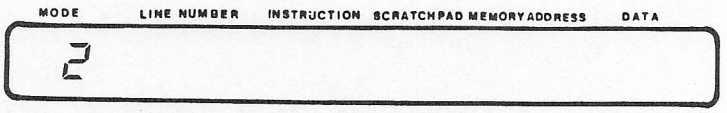
[ENTER]

(data display updated)



Press STOP key to end three-digit SMA monitoring:

[STOP]



A two-decade counter or timer can be monitored in a similar way. However, it should be borne in mind that the programming unit is organized here to monitor three consecutive scratchpad places. So only two of the three digits displayed will be of immediate importance. Steps 1 and 2 given above must still be observed.

4.4.4 Changing program words

Program words can be changed on-line by using the CHA(NGE) key. Proceed as follows:

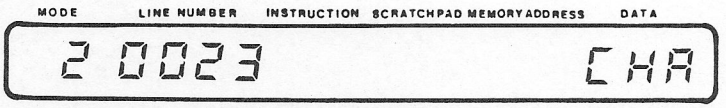
1. Set key switch to ON position.
2. Depress CHA key. The abbreviation 'CHA' will appear in the DATA field.
3. Depress LINE key and type in the line number *including* any leading zeros, to shift the existing line number out of the LINE display. So for line number 23 enter 0023.
4. Type in new program word (instruction part and address part) and check display.
5. Depress ENTER key. The acronym 'EOJ' (End Of Job) will appear in the DATA field as soon as the job is completed while the new program word will be fed back to the display.

When, in step 5, the ENTER key is depressed, the new program word will be written in the RAM program memory on the selected address – that is, the line number entered in step 3 – during the UDC phase (PC20 system in its normal, running condition).

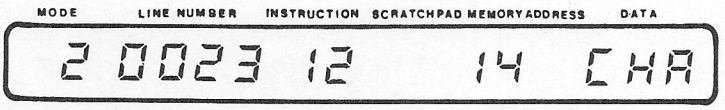
The program of Section 4.4.1 is again taken as an example. We want to change the program word on line 23, reading 'FTCHC 15' into a new word: 'FTCHC 14'. This is worked out in the following example.

Example 4-16 Changing a program word

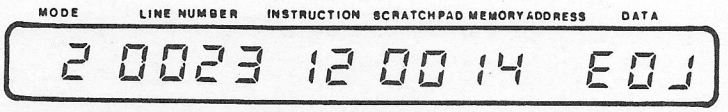
Depress CHA key and type in line no.:
[CHA] [LINE] [0] [0] [2] [3]



Type in new program word 'FTCHC 14':
[FTCHC] [1] [4]



Check display and enter word:
[ENTER] Job completed ——————►



4.4.5 Entering data into the scratchpad memory

Four-bit data can be entered into the scratchpad by using the CHA(NGE) and DATA keys. Thus, input signals and data sources not yet available may be imitated for program testing. The data are typed in on the digit keyboard as a numeral; these are converted internally into binary code for write-in on the selected four-bit location. See Table 4-7.

TABLE 4-7 Conversion of binary into decimal code.

one-bit scratchpad address				numeral to be typed in
.3	.2	.1	.0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

For entering data acts as follows:

1. Set key switch to its ON position.
2. Depress CHA key. The abbreviation 'CHA' will appear in the DATA field.
3. Depress SMA key and type in the four-bit scratchpad address where the data are to be entered.
4. Depress DATA key and type in the four-bit data to be entered as a numeral on the digit keyboard; see Table 4-7. Check display.
5. Depress ENTER key. The word 'EOJ' will be displayed in the DATA field when the job is completed.

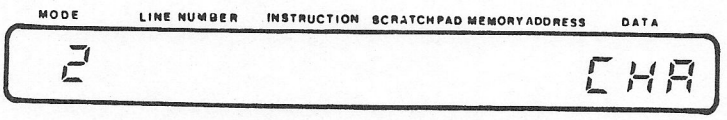
When the ENTER key is operated, the data will be written in the four-bit scratchpad address (selected in step 3) during the UDC phase (PC20 system in running condition).

It is shown in the following example how a set-point value can be written in. It is assumed that set-point numeral 98 must be entered on scratchpad locations 500 and 501, respectively.

Example 4-17 Entering data into the scratchpad memory.

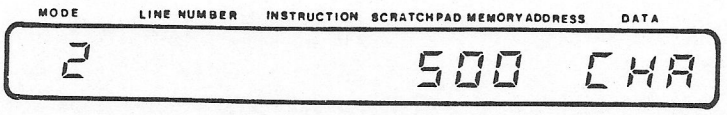
Depress CHA key:

[CHA]



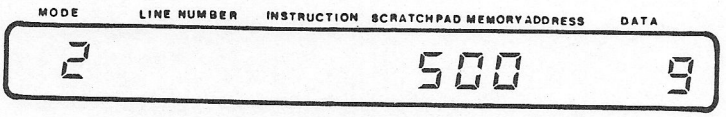
Select scratchpad address:

[SMA] [5] [0] [0]



Depress DATA key and type in data:

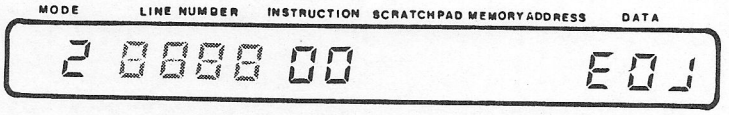
[DATA] [9]



Check display then write in:

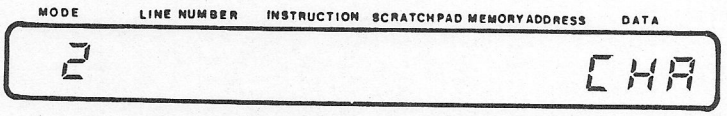
[ENTER] Job completed ——→

(zeros have no significance)

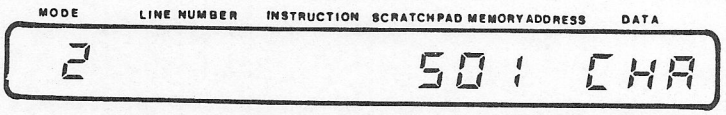


Follow same procedure for scratchpad location 501:

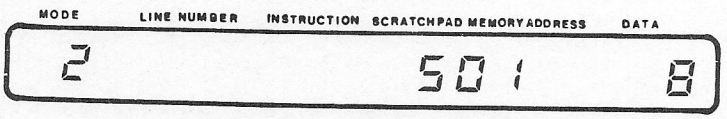
[CHA]



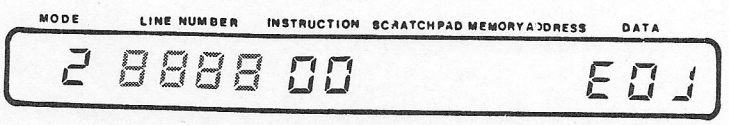
[SMA] [5] [0] [1]



[DATA] [8]



[ENTER]



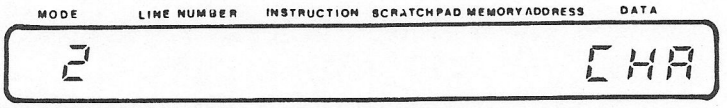
The contents of locations 500 and 501 can be checked through SMA monitoring as discussed in Section 4.4.2.

It will now be shown how a *single bit* in the scratchpad can be altered although only four-bit data can be written in. It is assumed that the contents of the four-bit address no. 200 is 1001 and that we want to change the state of address 200.2 to '1' level. We now have to type in the equivalent of 1101, which, as seen from Table 4-7, accords with numeral 13. This is worked out in Example 4-18.

Example 4-18 Changing single-bit data.

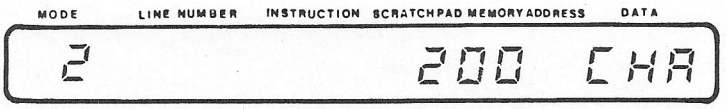
Depress CHA key:

[CHA]



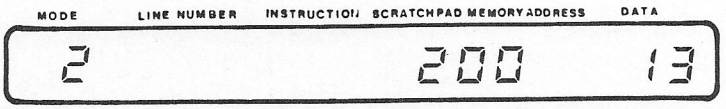
Select scratchpad address:

[SMA] [2] [0] [0]



Depress DATA key and type in data:

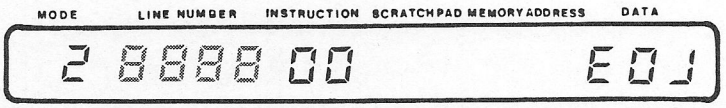
[DATA] [1] [3]



Check display, then enter:

[ENTER] Job completed ——————→

(zeros have no significance)



Again, the contents of scratchpad location 200 can be observed through SMA monitoring; the letter '.d' – for 13 (Table 4-5, Section 4.4) – will then appear in the left-hand part of the DATA field.

4.4.6 The result register

The result register included in the logic analyzer consists of a single flip-flop. The logic analyzer, as its name implies, performs logic operations in response to logic instructions and stores the outcome in the result register. When the first of a series of logic instructions is met, the contents of the result register will be made to assume the value, 0 or 1, of the addressed variable. For any subsequent logic instruction, the contents of the result register are processed, together with the value of the new variable, in accordance with that instruction and the new result is written in the result register. Thus, any intermediate result is stored in the result register, until the last logic instruction in the series has been processed. Then, any other type of instruction will not change the contents of the result register because these are not processed by the logic analyzer.

Note, further logic instructions that do not, in a given instance, affect the result of previous logic operations are not processed. Consequently, they will not change the contents of the result register. This may occur in the case of a series of AND instructions followed by an OR instruction, thus:

A.B.C + D.E.F + ...

If the result of the first term is '1', the following terms will be of no consequence and, as a result, will be disregarded by the logic analyzer.

The result register is extremely useful to check if logic processing occurs as prescribed. One example is given here to clarify the matter; the condition of the result register is indicated by the 'RR' LED on the PU20 panel. Note that, as explained above, for each new line the obtained contents, (RR), of the result register are processed together with the new variable in accordance with the logic instruction on that line. Because the logic result, '1', on line 106 cannot change any more, the subsequent logic instructions (lines 107 to 109) will be ignored. Further, the instructions on lines 111 to 113 are not seen by the logic analyzer and so will leave the contents of the result unaltered.

Example 4-19 Reading out the result register.

line	prog. word		state of addressed scratchpad location	logic operation	(RR)	'RR' LED
:	:					
0100	EQL	8.0	X		X	X
0101	AND	4.0	1	(RR) = (4.0) = 1		ON
0102	ANDNT	4.1	1	(RR) . $\overline{(4.1)}$ = 1 . 0 = 0		OFF
0103	ORNT	4.2	0	(RR)+$\overline{(4.2)}$ = 0 + 1 = 1		ON
0104	AND	4.3	0	(RR) . (4.3) = 1 . 0 = 0		OFF
0105	OR	5.0	1	(RR)+(5.0) = 0 + 1 = 1		ON
0106	ANDNT	5.1	0	(RR) . $\overline{(5.1)}$ = 1 . 1 = 1		ON
0107	OR	5.2	1	These instructions	1	ON
0108	AND	5.3	0	ignored in this	1	ON
0109	ANDNT	6.0	0	instance	1	ON
0110	EQL	8.1	1		1	ON
0111	JFRF	3		These instructions	1	ON
0112	FTCHC	10		not seen by	1	ON
0113	STRD	103		logic analyzer	1	ON
0114	AND	10.0	0		0	OFF

X = Don't care.

4.5 Monitoring cycle-by-cycle

The mode 'monitoring cycle-by-cycle' is helpful in examining specific program functions in detail while the program is still in its design stage. Transient phenomena, such as active one-shot outputs, might pass unobserved owing to display delay when the PC20 operates in its continuous mode. The performance of such functions can be examined in the monitoring cycle-by-cycle mode because the central processor of the PC20 system is arrested by the PU20 in its UDC (UpDate and Check) phase to perform single cycles on command, that is, whenever the ENTER key is depressed.

With the key switch turned on, all functions described before, with the exception of three-digit SMA monitoring, will become available, that is:

- Line monitoring.
- SMA monitoring.
- Changing program words.
- Entering data into the scratchpad memory.

The procedures dealt with in Section 4.4 are followed to carry out all of the above functions. Note that in the monitoring cycle-by-cycle mode the S^- key cannot be used; this means that any line number lower than that on display must be chosen using the digit keyboard (after depressing the LINE key). Any higher line number can be selected and its contents displayed by operating the S^- key, then the ENTER key.

It should be borne in mind that timers, counters and one-shots driven by an input signal not related to the PC20 repetition period do not perform here as prescribed in the program. To test their operation, the input signal must be temporarily replaced by one that does not change more than once per PC20 cycle.

Fig. 4-20 shows clearly the operation of the PC20 system in the monitoring cycle-by-cycle mode for the case where output 200.3 must follow input 100.3 (system halted in UDC phase). Each time the ENTER key is operated, the system will make one complete cycle, then wait again in the UDC phase. Input 100.3 is now changed from its inactive state to its active state. It is seen that *two* PC20 cycles are needed (ENTER key depressed twice) for the output to assume '1' level: *(cycle 1)* writing the input condition in the scratchpad, *(cycle 2)* processing the new input condition and setting the output latch. Similarly, the PC20 must make two complete cycles for the output to return to '0' level when the input is made inactive again.

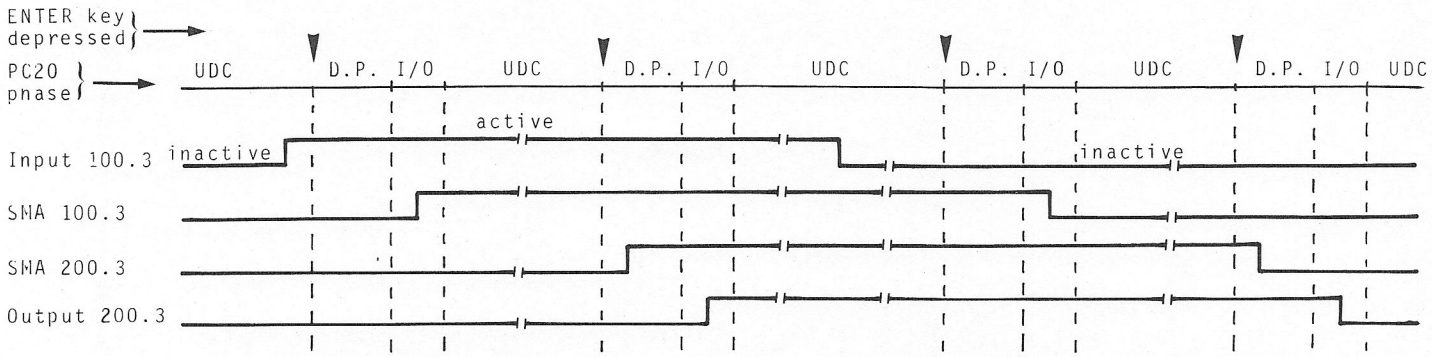


Fig. 4-20 Cycle-by-cycle operation of PC20 system. Note that ENTER key must be depressed twice for output 200.3 to follow input 100.3. UDC = UDC phase; D.P. = Data processing phase; I/O = input/output phase. Reset phase not shown for simplicity.

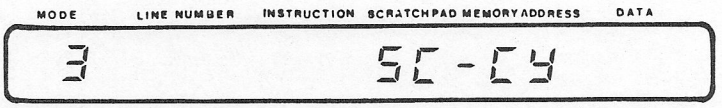
The situation of Fig. 4-20 can be expressed in this simple program part:

line	program word		
0100	AND	100.3	If input 100.3 is activated
0101	EQL	200.3	output 200.3 will become HIGH.

In the following example it is shown how the change of output 200.3 is observed (see Fig. 4-20).

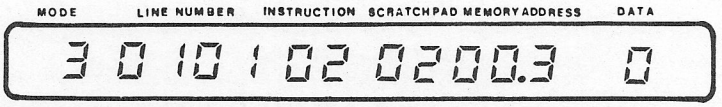
Example 4-20 Change of output in response to input.

Monitoring cycle-by-cycle mode selected (see Section 4.2.2)



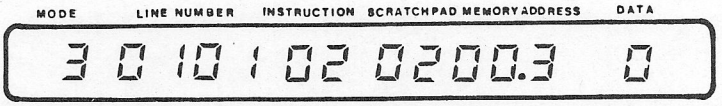
Select line 101:

[LINE] [1] [0] [1] [ENTER]



Make input 100.3 active, then depress ENTER key 1st time:

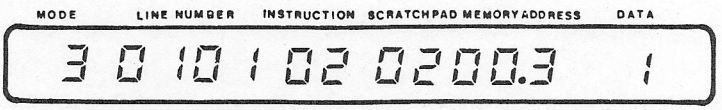
[ENTER]



Depress ENTER key 2nd time:

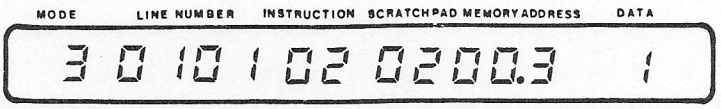
[ENTER]

(LED of output 200.3 will now be lit)



Make input 100.3 inactive, then depress ENTER key 1st time:

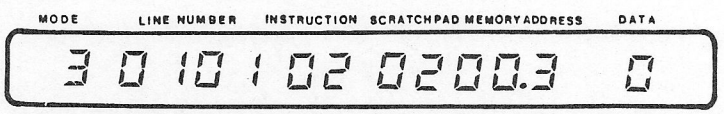
[ENTER]



Depress ENTER key 2nd time:

[ENTER]

(LED of output 200.3 will now extinguish)



4.6 Monitoring step-by-step

In the previous mode the change in contents of a given scratchpad location can be observed while the program is being executed, each time the system performs one complete cycle. In the mode to be described here even more accurate program checking is possible because the PC20 is made to step through its data processing phase at a speed determined by the user. Meanwhile, execute instructions will be obeyed if permitted by prevailing logic input conditions. Thus, any programming error or a source of faulty operation can be more easily traced. In this way we can check all or part of the program. When the LSTIO and END instructions are met, the system will pass though the I/O phase at its normal speed and wait at the start (line number 0000) of the next data processing phase. It will also be seen that JUMP instructions are carried out provided that their input conditions are satisfied.

Once the monitoring step-by-step mode has been selected and the ENTER key depressed, the programming unit will halt the central processor in the data processing phase. Each time the S^+ key is depressed, the next program word and the appertaining scratchpad data will be shown. *Depressing the S^+ key means here that execute instructions will or will not be carried out in accordance with programmed input conditions.*

The information in Section 4.5 about timing circuits also applies for this mode of PC20 operation.

To examine a particular program section proceed as follows:

1. Turn key switch to ON position.
2. Select monitoring step-by-step mode (Section 4.2.2).
3. Depress LINE key and type in on digit keyboard the line number where program checking is to start.
4. Depress ENTER key. Program word and processed scratchpad contents will now be displayed.
5. Depress S^+ key to display next program word and processed scratchpad contents.
6. Repeat step 5 for subsequent line numbers, etc.

Only line monitoring, described above, is possible in the monitoring step-by-step mode. Further, the S^-key must not be operated in this mode.

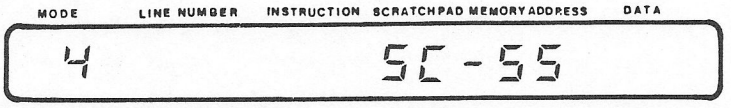
To clarify what has been discussed, the following simple program will be worked out in an example:

line	progr. word		
0000	AND	4.0	If 4.0 is LOW
0001	JFRF	3	Jump 3 lines forward
0002	AND	4.1	(If 4.1 is HIGH
0003	EQL	8.0	8.0 will be HIGH.)
0004	LSTIO	11	Last address in I/O phase is SMA 11.
0005	END	4	First address in I/O phase is SMA 4.

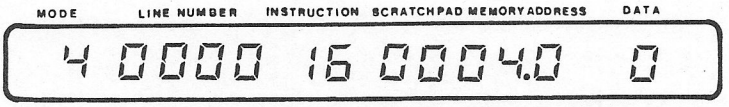
It is assumed that the contents of locations 4 and 11 in the scratchpad are 'zero'; because the state of 4.0 is '0', the JFRF instruction on line 0001 will be executed.

Example 4-21 Illustration of monitoring step-by-step mode.

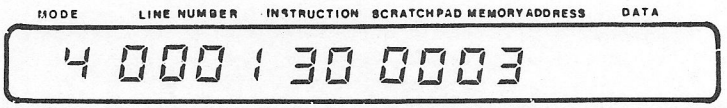
Monitoring step-by-step mode selected
(see Section 4.2.2)



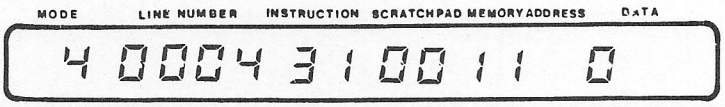
Select first line no.:
[LINE] [0] [ENTER]



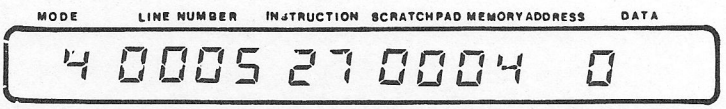
Depress S^+ key:
[S^+]



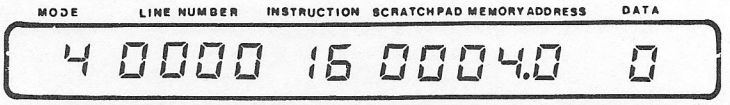
Depress S^+ key:
[S^+] (JFRF executed)



Depress S^+ key:
[S^+]



Depress S^+ key:
[S^+]
(system returns to start of program)



4.7 EPROM programming

In this mode the PC20 program is dumped into EPROMs placed in the sockets on the programming unit. Verification occurs during dumping. That is, each program word written in the EPROMs is verified against the original program word in the PC20 program memory. Program dumping will stop as soon as an error is detected.

Dumping occurs in program blocks of either 1k or 2k (multiples of 1024 program words). It starts at address 0000 in the EPROMs but the start address in the PC20 program memory can be freely chosen. The end of successful dumping is indicated by the word 'EOJ' in the DATA field.

To accept the program, EPROMs of either 1k or 2k memory capacity are used, such as the 2758 (1k8) or 2716 (2k8). The EPROMs must be properly marked before insertion because each EPROM can take only half of each sixteen-bit program word (see Section 4.13.3).

Time for dumping is 50 ms for each program word or 51 s for 1k program length. If 1k program length is chosen and the EPROMs have 2k capacity, only half of the EPROMs will be filled with program.

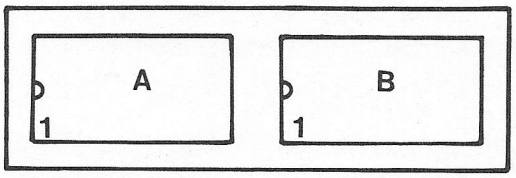


Fig. 4-21 Marking of EPROMs.

For EPROM programming the following steps are taken:

1. Mark EPROMs (Fig. 4-21) and insert in sockets on programming unit. The EPROMs are clamped in position by lever action.
2. Set key switch to ON position.
3. Select EPROM programming mode (Section 4.2.2).
4. Depress LINE key and type in line number of program memory in PC20 at which EPROM programming is to start, then depress ENTER key; 'RUN' LED will light.
5. Depress DATA key and select length of program to be dumped by typing in numeral '1' on digit keyboard for 1024 words of program length (1k) or numeral '2' for 2048 words of program length (2k). Now depress ENTER key.
6. Depress START key. 'BUSY' LED will light while dumping starts simultaneously with verification. At the end of program dumping, 'BUSY' and 'RUN' LEDs will extinguish and the word 'EOJ' will appear in the DATA field.
7. Remove EPROMs from sockets.

If an error occurs during dumping, the 'ERROR' LED lights and code E12 (verify error; see Table 4-1, Section 4.2.3) appears in the DATA field. The display now shows the original program word and its address (line number) in the PC20 program memory. By depressing the CHA key, the program word written in the EPROMs will be displayed but the line number of the original program in the PC20 memory will remain. The STOP key must be depressed and the EPROMs cleared before programming can recommence.

Fig. 4-22 shows how the PC20 program is copied in the EPROMs. It is assumed that the 2k program memory is filled with program and that the EPROMs have 2k memory capacity. The program to be dumped runs from line 50 to line 1947 so we have to select 2k program length on the digit keyboard. Because 2048 words of program are dumped, the start of program, lines 0 to 49, will appear on the last fifty addresses in the EPROMs. However, this will have no harmful effect owing to the LSTIO and END instructions placed on the highest addresses in the PC20 program memory. If the original program length were 3k (written in a 4k memory), the program part line numbers 50 to 2097 would be dumped.

Fig. 4-22 is elaborated in the example that follows here.

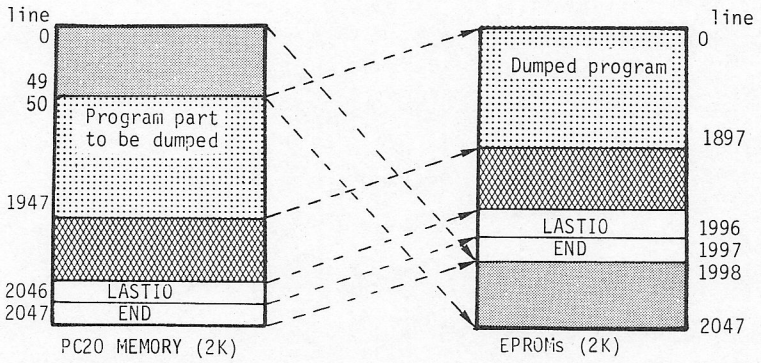
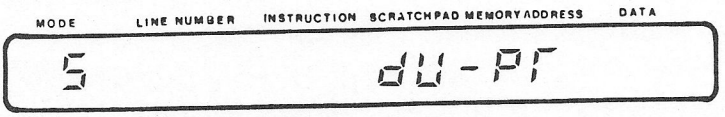


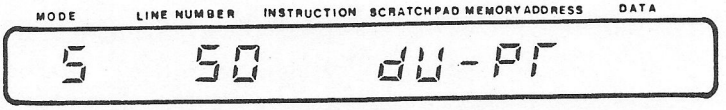
Fig. 4-22 Transfer of PC20 program to EPROMs, starting at line 50. Note that beginning of program is dumped on highest addresses in the EPROMs.

Example 4-22 EPROM programming.

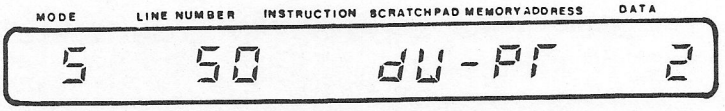
EPROM programming mode selected
(see Section 4.2.2)



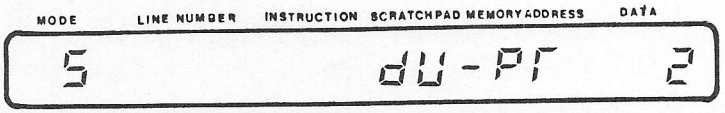
Select start line no.:
[LINE] [5] [0] [ENTER]



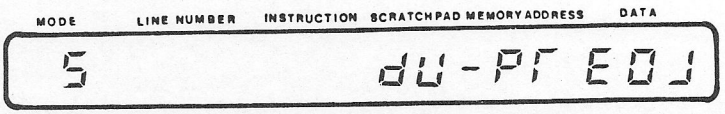
Select length of program to be dumped:
[DATA] [2] [ENTER]



Start EPROM programming:
[START]



Programming completed ——————→



4.8 Dump on cassette

To dump the PC20 program on cassette any high-quality domestic cassette recorder can be used; the specifications that are to be met are detailed in Section 4.13. C60 super-quality FeCr cassette tape is recommended to avoid drop-outs and undue tape stretching. The program is copied on both stereo tracks. A 180°, five-pins DIN connector is available at the rear of the programming unit; for pinning see Section 4.13.

Baud rate is 600 baud. Dumping time is 6 min per 1k program length (0,35 s for each program word). One side of a C60 compact cassette will take 5k length of program. Fig. 4-23 shows how a program word complete with its line number is dumped on magnetic tape (serial read-out); the parity bit is even. The '0' and '1' logic bit levels will be reproduced as 0 kHz and 5 kHz signals, respectively. Each transmitted program word contains nineteen, eleven-bit bytes.

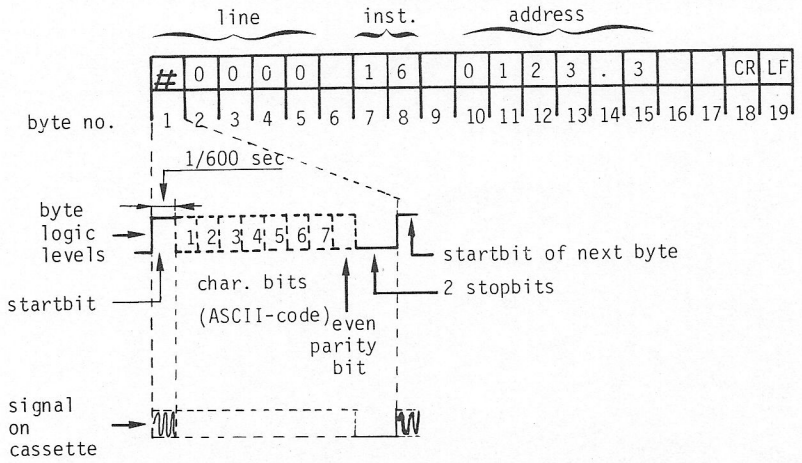


Fig. 4-23 Organization of transmitted program word.
State '0' recorded as 0 kHz; state '1' recorded as 5 kHz.

Dumping is carried out in 1k blocks of program (1024 program words per block), from 1k to 8k program length; it should be remembered, however, that one side of a C60 cassette cannot take more than 5k of program length. The user is at liberty to choose the line number where dumping is to start. By operating the STOP key, dumping may be terminated at any desired line number; the recorded program can then be used with the inclusion of the program word dumped last before the STOP key has been operated.

Both at the beginning and end of dumping, rub-outs will be generated during 4 s, to allow the tape to assume speed and permit space for the start of a second program to be dumped on the same cassette side. Dumping terminated, the word 'EOJ' will appear in the DATA field.

To check the accuracy of the dumped program, the verify facility of Mode 9 must be used (see Section 4.11).

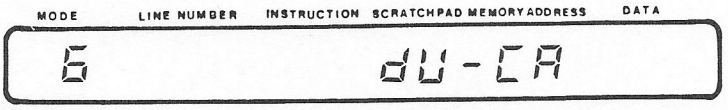
For dumping on cassette the following steps are taken:

1. Connect cassette recorder to PU20.
2. Set recording and treble controls to maximum and bass control, if provided, to minimum. Volume control may be turned up for monitoring. Reset counter. Make sure that the tape is rewound and that the recording mode has been selected.
3. Set key switch in ON position.
4. Select mode 'Dump on cassette' (Section 4.2.2).
5. Depress LINE key and type in line number of program in PC20 program memory where dumping is to start, then depress ENTER key.
6. Depress DATA key and select length of program to be dumped by typing in numeral on digit keyboard: numeral 1 for 1k program length to numeral 5 for 5k program length and C60 cassette (multiples of 1024 program words). Now depress ENTER key.
7. Start recorder.
8. When tape has assumed running speed, press START key; 'BUSY' and 'RUN' LEDs will become illuminated. Rub-outs will be generated during 4 s while the word 'GAP' is shown in the DATA field. When this word disappears, dumping will start and the successive line numbers appear in the LINE field. At the end of dumping, rub-outs are again produced during 4 s ('GAP' in DATA field).
9. When 'EOJ' appears in DATA field ('BUSY' and 'RUN' LEDs extinguish) stop recorder and note down its counter position. It will thus be easy to find end of program for the case where a second program is to added on the same cassette side.

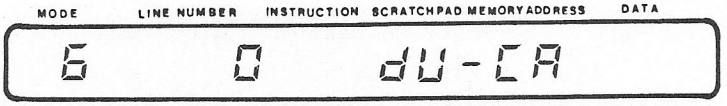
The above exercise will be illustrated in an example. It is assumed that we want to dump a 3k program starting at line 0000.

Example 4-23 Dump on cassette.

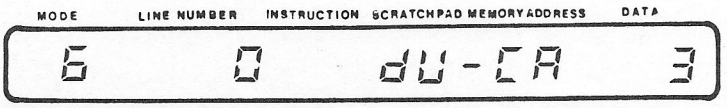
Mode 'Dump on cassette selected
(see Section 4.2.2)



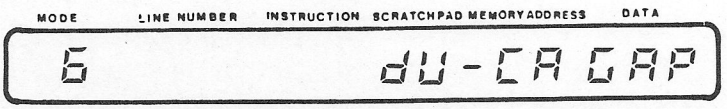
Select start line no.:
[LINE] [0] [ENTER]



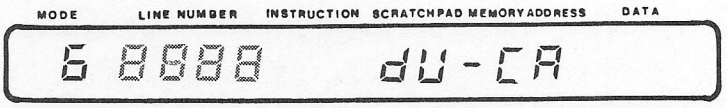
Select length of program to be dumped:
[DATA] [3] [ENTER]



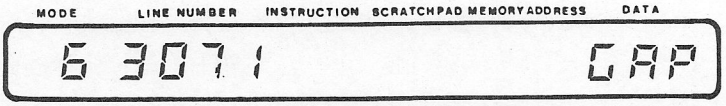
Start recorder then depress START key:
[START]



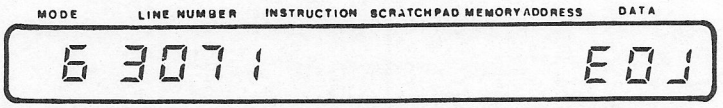
Dumping proceeds ——————→
(line numbers shown)



End of dumping ——————→
(display shows last loaded line no.)



Job completed ——————→



4.9 Dump on RS449/423

The PU20 is provided with an RS449/423 connector to communicate with punchers, readers, printers, VDU terminals and higher-intelligence peripherals. The signals on this connector conform to the EIA-standard RS449/423 data transmission configuration DT (Data and Timing only). For specifications refer to Section 4.13.

The information to be given to the programming unit must contain the baud rate, parity and format of the program to be transmitted. The parameters to be typed in are specified in Table 4-8. The formats coded 'C4' and 'C5' are useful for filing. Fig. 4-24 shows examples of print-outs.

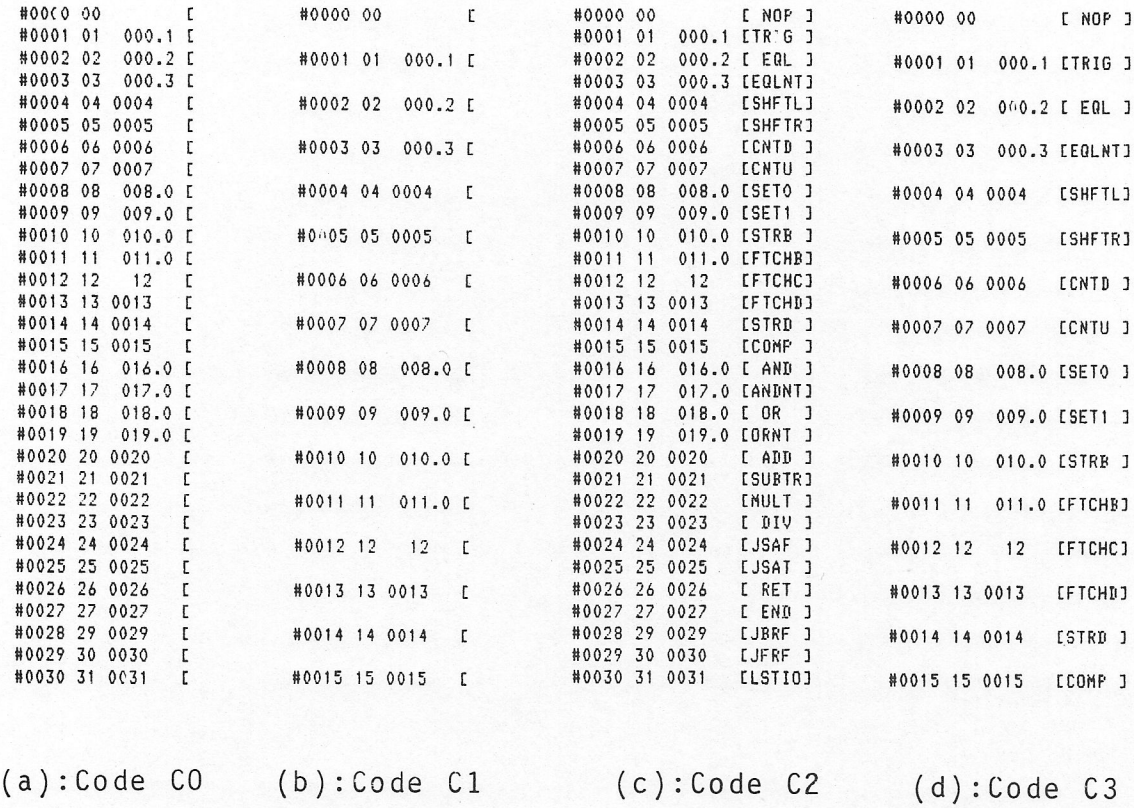


Fig. 4-24 Examples of hard copy:
(a) no mnemonics, 1 × LF;
(b) no mnemonics, 2 × LF;
(c) with mnemonics, 1 × LF;
(d) with mnemonics, 2 × LF.

TABLE 4-8 Parameters to be entered on digit keyboard.

baud rate	parity	format
b0 = 110 baud	P0 = even	C0 = no mnemonics, 1 × LF
b1 = 150 baud	P1 = odd	C1 = no mnemonics, 2 × LF
b2 = 300 baud	P2 = don't care	C2 = with mnemonics, 1 × LF
b3 = 600 baud		C3 = with mnemonics, 2 × LF
b4 = 1200 baud		C4 = with mnemonics, 1 × LF, A4 format, 50 lines per page
		C5 = with mnemonics, 2 × LF, A4 format, 25 lines per page

Dumping occurs in 1k program blocks, from 1k to 8k program length, starting at a line number which can be chosen at liberty. The flow of data is halted by using the STOP key (communication maintained); the PU20 will then stop after transmission of the program word in transit has been completed. To restart the data flow, the DATA key must be depressed. This mode of operation is especially useful to examine particular program sections on the VDU screen. As in the mode 'Dump on cassette', rub-outs are generated during 4 s both at the beginning and end of dumping. Program dumping can be terminated at any desired line number by *twice* depressing the STOP key. This will terminate communication between the PU20 and peripheral equipment.

Verification does not occur during program dumping. To check the accuracy of, for instance, a punched copy, the verify facility of Mode 10 must be used (Section 4.12).

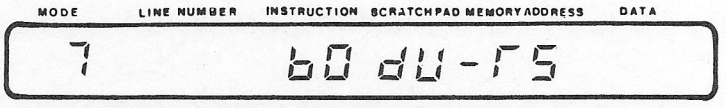
The procedure for dump on RS449/423 runs as follows:

1. Connect peripheral to programming unit.
2. Turn key switch to ON position.
3. Select mode 'Dump on RS' (Section 4.2.2). Code 'b0' will now appear in the INST field.
4. Type in required baud rate on digit keyboard (Table 4-8). If 1200 baud must be selected, numberal 4 is entered, which will change code 'b0' into 'b4'. If 110 baud is needed a numeral need not be entered and code 'b0' will stay.
5. Depress ENTER key, which will cause code 'PO' to appear in the INST field.
6. Type in desired parity on digit keyboard (Table 4-8). If an odd parity is desired, enter numeral 1; this changes code 'PO' into 'P1'. If an even parity is required, a numeral need not be entered and code 'P0' will stay.
7. Depress ENTER key. Now code 'C0' will be displayed in the INST field.
8. Type in desired format on digit keyboard (Table 4-8). If program must be dumped in A4 format, 50 lines per page, enter numeral 4; code 'C0' on display will change to code 'C4'. If a continuous print-out with no mnemonics and 1 x LF is required, a numeral need not be entered and code 'C0' will remain.
9. Depress ENTER key. Code in INST field will now disappear to indicate that all parameters have been entered.
10. Depress LINE key and type in line number of PC20 program where dumping is to start, then depress ENTER key.
11. Depress DATA key and select length of program to be dumped by typing in numeral 1 for 1k program length to numeral 8 for 8k program length (multiples of 1024 program words). Now depress ENTER key.
12. Start peripheral.
13. Depress START key. 'BUSY' and 'RUN' LEDs light and rub-outs will now be transmitted during 4 s while the word 'GAP' is displayed in the DATA field. When this word disappears, dumping will start and the successive line numbers of the dumped program will appear in the display. At the end of dumping, rub-outs will again occur during 4 s ('GAP' in DATA field).
14. When 'EOJ' appears in DATA field ('BUSY' and 'RUN' LEDs extinguish) the job will be ready. Now stop peripheral.

The above procedure will be worked out in an example here. It is assumed that we want to make a hard copy of 1550 lines of program on TTY, so we have to select 2k program length (2048 words). (STOP key can be operated when 1550 words have been printed out.) Baud rate is 110 baud (code 'b0') and parity checking is odd (code 'P1'). For filing, format A4 with fifty lines per page is chosen (code 'C4'). Dumping must start at line 0.

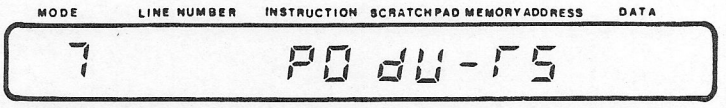
Example 4-24 Dump on RS449/423.

Mode 'Dump on RS' selected
(see Section 4.2.2)

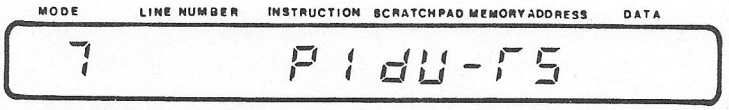


Required baud rate is 110 baud;
depress ENTER key:

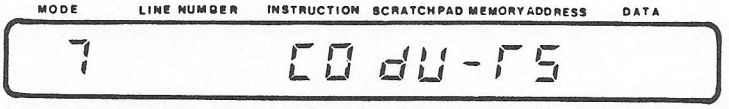
ENTER



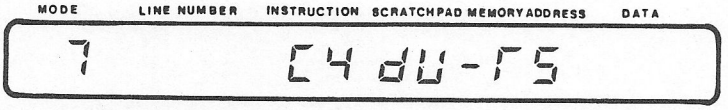
Select odd parity checking:
[1]



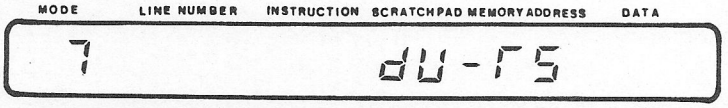
Depress ENTER key:
[ENTER]



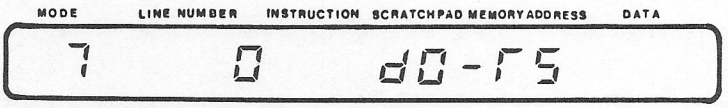
Select format:
[4]



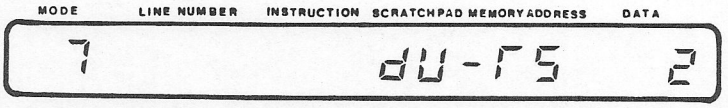
Depress ENTER key:
[ENTER]



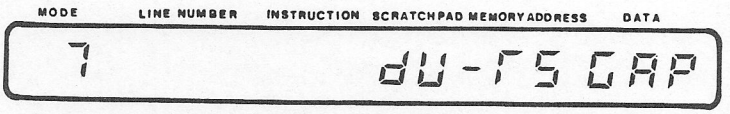
Select start line no.:
[LINE] [0] [ENTER]



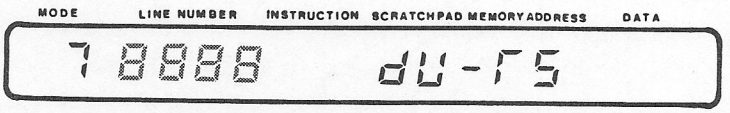
Select length of program to be dumped:
[DATA] [2] [ENTER]



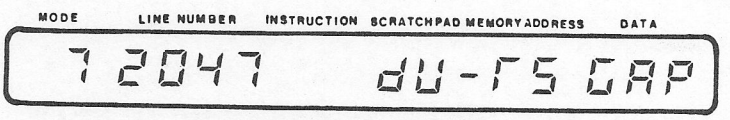
Now start peripheral then depress START key:
[START]



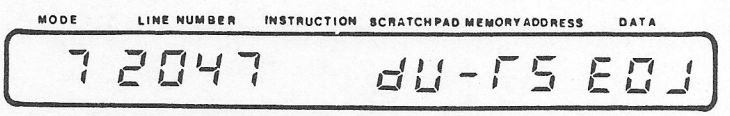
Dumping proceeds ——————→
(line nos. shown)



End of dumping ——————→



Job completed ——————→
(display shows last loaded line no.)



4.10 Load from EPROMs

The procedure to be followed by the user for the mode 'Load from EPROMs' is similar to that for the EPROM programming mode, Section 4.7 However, the program data now flow in the reverse direction, that is, from the EPROMs inserted in the sockets on the PU20 to the PC20 program memory. This implies that a C-MOS RAM must be included in the PC20 system. This mode is useful where a program must be changed or extended. The existing program stored in the EPROMs is entered into the C-MOS RAM, changes and extensions are made, then the new program can be again dumped in the EPROMs as described in Section 4.7. See Section 4.13.3 for storage of program words in EPROMs.

Loading occurs in program blocks of either 1k or 2k. The EPROM start address is always 0 but dumping into the RAM may start at any desired line number. Load time is 15 ms per program word or 15 s per 1k block of program. End of loading is indicated by the word 'EOJ' in the DATA field.

As in the EPROM programming mode, verification occurs simultaneously with loading. If an error occurs, loading stops and the display shows the program word stored in the RAM together with its RAM address (line number). Also, the 'ERROR' LED lights and code E12 appears in the DATA field. The original program word present in the EPROMs is shown when the CHA key is depressed (RAM line number staying on display). After the STOP key has been operated, loading can be re-started.

The steps to be taken are similar to those described in Section 4.7, only the mode 'Load from EPROMs' must now be selected (Section 4.2.2).

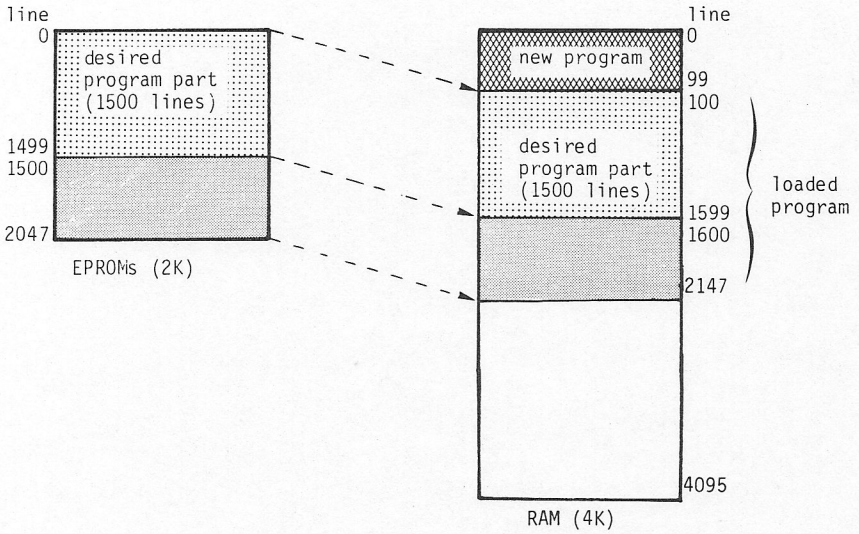


Fig. 4-25 Transferring 1500 lines of program from EPROMs to RAM.

Fig. 4-25 illustrates how the program will be copied in a 4k RAM for the case where a new program has been written on lines 0 to 99 and so loading is to start at line 100. If we want to copy 1500 lines of program we have to load a 2k block of program, which will appear on lines 100 to 2147. Because only 1500 lines of the dumped program are wanted, we will continue programming on line 1600. However, care must be exercized if the RAM capacity is exceeded during loading, see Fig. 4-26. Fig. 4-26 is identical to Fig. 4-25, with the exception that now 2500 lines have already been programmed. Because a 2k block of program is dumped, the first 452 lines of new program written in the RAM are overruled by the last lines of the unwanted program section in the EPROMs. *Therefore, if the highest RAM address will be exceeded during dumping, dump first then enter the new program into the RAM.* Now the unwanted part of the EPROM program appearing on the lowest RAM addresses will be overwritten by the new program.

The case shown in Fig. 4-25 is used in the following example.

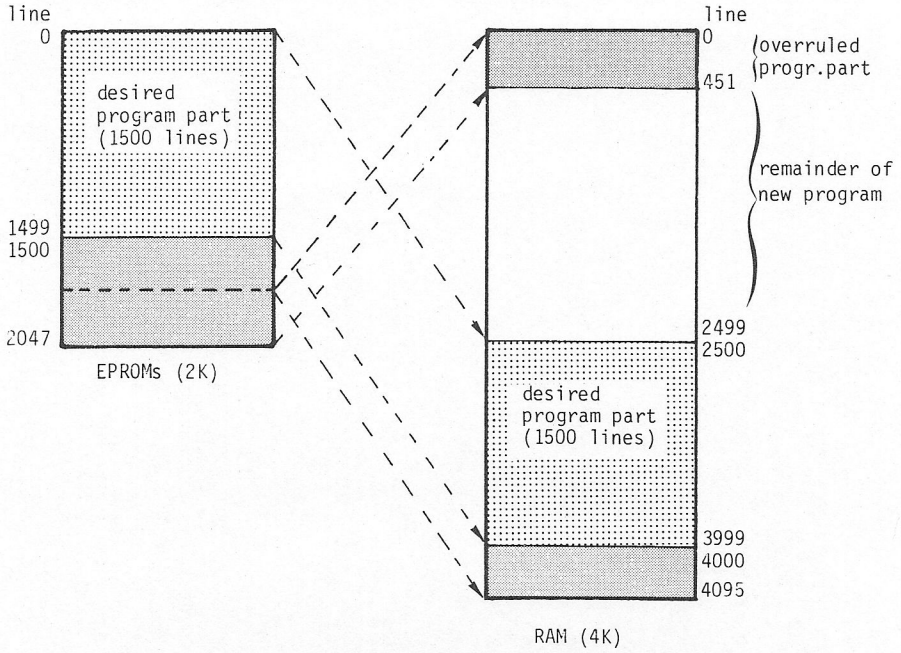
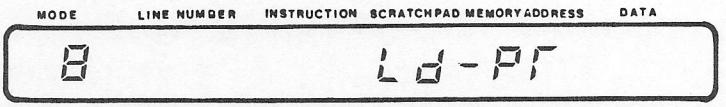


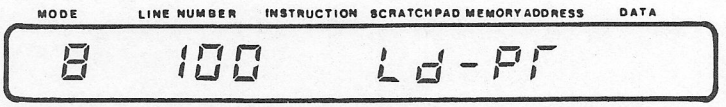
Fig. 4-26 Transferring program from EPROMs to RAM, where the RAM capacity is exceeded. Note that end of EPROM program appears on first addresses of RAM.

Example 4-25 Load from EPROMs.

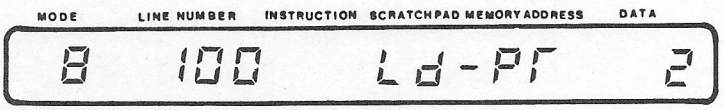
Mode 'Load from EPROMs' selected (see Section 4.2.2)



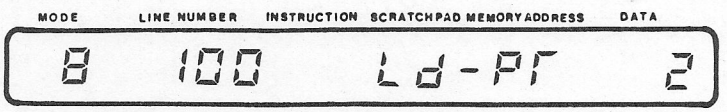
Select start line no.:
[LINE] [1] [0] [0] [ENTER]



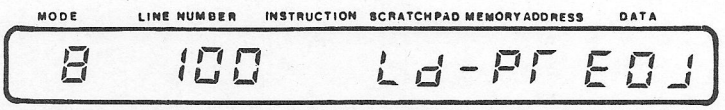
Select length of program to be loaded:
[DATA] [2] [ENTER]



Start EPROM programming:
[START]



Loading completed ——————→



4.11 Load from cassette

In this mode the program dumped on an audio cassette is loaded into the C-MOS RAM of the PC20 system. Here, the data will flow in the reverse direction to that for the mode 'Dump on cassette'; see Section 4.8. The mode 'Load from cassette' is also used to verify a program just loaded into the C-MOS RAM or previously dumped on cassette.

For this mode, the start address on cassette and in the RAM must both be selected. Loading occurs in 1k blocks of program (multiples of 1024 program words), from 1k to 8k; however, not more than 5k can originate from one side of a C60 cassette. As in the case of dumping, transmitting speed is 600 baud (serial read-in from cassette) so load time is 6 min per 1k program length.

Fig. 4-27 shows how the program is transferred from cassette to RAM for 1k length of program to be loaded and starting at cassette address 100. It is assumed that 49 words of new program have been entered on the first RAM addresses so loading must start at line 50 in the program memory. If, however, dumping were to start at, say, line 3500 of a 4k memory, the last part of dumped program would be written on the lowest RAM addresses, similarly to the case of Fig. 4-26. *Therefore, if the highest RAM address will be exceeded during dumping, dump first then enter new program into RAM.*

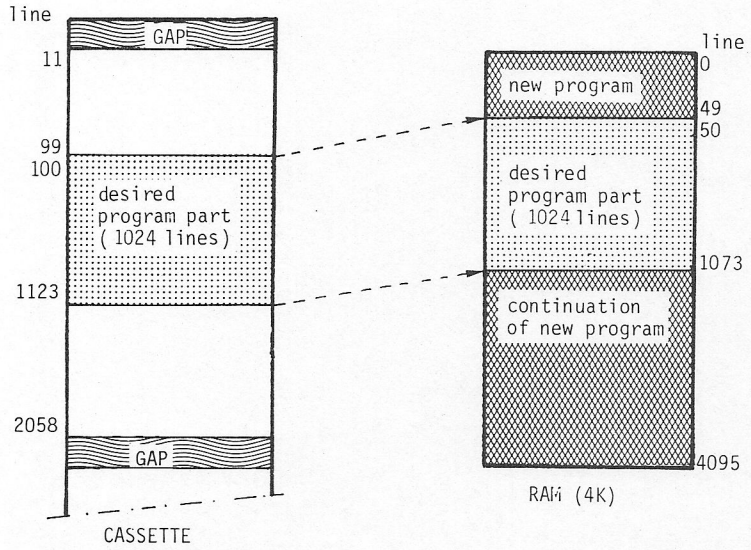


Fig. 4-27 Transferring 1k of program from cassette to RAM.

As in the mode 'Dump on cassette' loading can be terminated at any desired line number by using the STOP key.

For *loading* act as follows:

1. Connect cassette recorder to PU20.
2. Select playback mode and turn playback control, if provided, to maximum. Find the position where run-in part (rub-outs) of program starts. Turning up volume control for monitoring will be helpful to find this position because the audio signal of run-in part is distinctly different from that of dumped program.
3. Turn key switch to ON position.
4. Select mode 'Load from cassette' (Section 4.2.2).
5. Depress LINE key and type in address of RAM where loading is to start, then depress ENTER key.
6. Depress SMA key and type in line number of program on cassette where loading is to start (first program word to be loaded), then depress ENTER key.
7. Depress DATA key and select length of program to be loaded by typing in numeral on digit keyboard: numeral 1 for 1k program length to numeral 5 for 5k program length and C60 cassette (multiples of 1024 program words). Now depress ENTER key.
8. Start recorder.
9. When tape is at operating speed, depress START key; 'RUN' LED lights. Cassette addresses appear in LINE field until address is found where loading is to start (step 6). Now 'BUSY' LED illuminates to indicate start of data transfer, and abbreviation 'Ld' is displayed in DATA field. Simultaneously the successive RAM addresses (line numbers) will now appear in LINE field.
10. Dumping ready, 'EOJ' will appear in the DATA field ('BUSY' and 'RUN' LEDs extinguish).
11. Stop cassette recorder.

It is seen that loading will occur if no further command is issued. However, the mode discussed here can also be used: (1) to verify the program just loaded into C-MOS RAM as described above, (2) to verify a previously dumped program. Section 4.8. For *verification,* one step must be interposed.

7a. Depress VFY key, then type in numeral 1, which will appear in the INST field. Now depress ENTER key.

In step 9, as soon as the cassette address appears where verification must start, 'VFY' will be displayed in the DATA field.

If, during verification, an error is detected, the PU20 will stop and display the program word of the RAM together with its line number. In addition, the 'ERROR' LED lights and the code 'E12' appears in the DATA field. When the CHA key is operated the program word on cassette is shown but the RAM line number will stand. Now, the STOP key must be operated before repeating the steps required for dumping (see Section 4.8) or loading (see above).

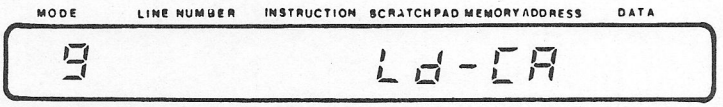
Now *to return to loading*, step 7a must be modified as follows:

7a. Depress VFY key, then type in numeral 0, which will appear in the INST field. Now operate ENTER key.

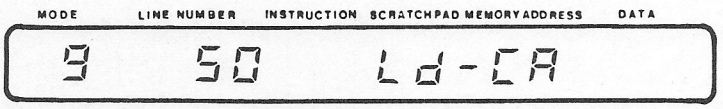
The case of Fig. 4-27 will be worked out here in an example. That is, start line number is 50 in RAM and start address is 100 on cassette.

Example 4-26 Load from cassette.

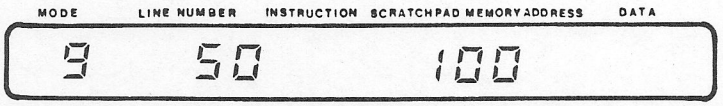
Mode 'Load from cassette' selected
(Section 4.2.2)



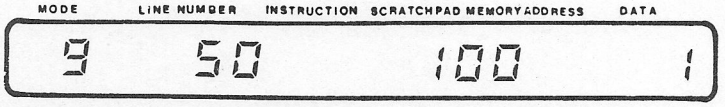
Select start line no. in RAM:
LINE 5 0 ENTER



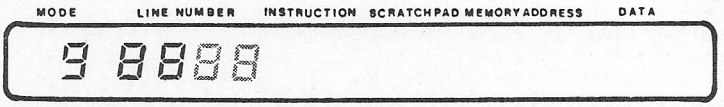
Select start address on cassette:
SMA 1 0 0 ENTER



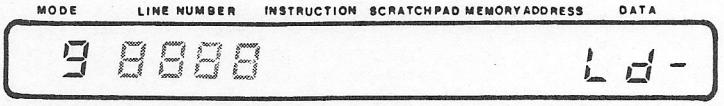
Select length of program to be loaded:
DATA 1 ENTER



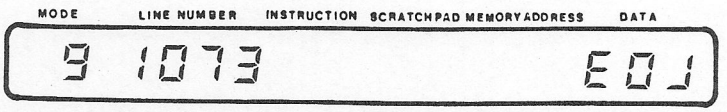
Start recorder then depress START key:
START
(cassette addresses shown)



Loading starts ——————→
(RAM addresses shown)



End of loading ——————→
(display shows last RAM address)



4.12 Load from RS449/423

The Mode 'Load from RS449/423' uses the route described in Section 4.9 but the data will be transferred here in the reverse direction, that is, from the external source to the C-MOS RAM in the PC20 system. This mode can also be used to verify a program either just loaded or previously dumped, for instance, via a puncher.

As in the 'Load from cassette' mode, the RS start address and that in the RAM must both be given. Loading can occur in blocks of program, from 1k to 8k. Transmitting speed may have a value between 110 baud and 1200 baud and parity may be even, odd or parity checking can be omitted; see Table 4-9 (cf. Table 4-8, Section 4.9).

TABLE 4-9 Parameters to be entered digit keyboard

baudrate	parity
b0 = 110 baud	P0 = even
b1 = 150 baud	P1 = odd
b2 = 300 baud	P2 = dont't care
b3 = 600 baud	
b4 = 1200 baud	

Fig. 4-28 shows how the program is stored in RAM. After what has been discussed before, no further explanation will be necessary here. *If RAM capacity is exceeded, dump first, then enter new program into RAM.*

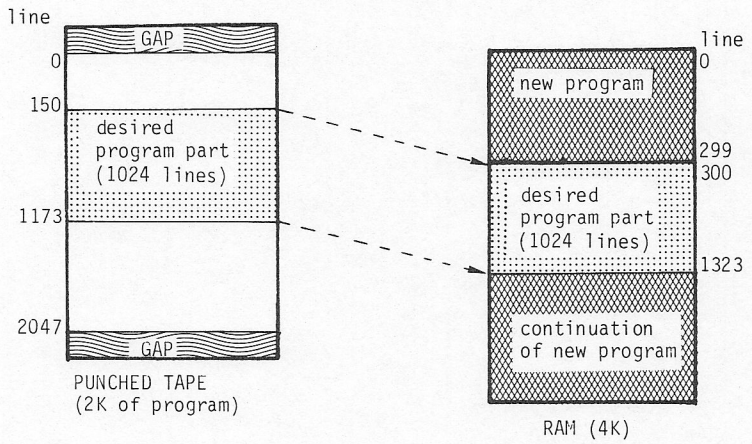


Fig. 4-28 Transferring 1k of program from peripheral to RAM.

The following steps apply for *loading:*

1. Connect peripheral to PU20 and position at the start of run-in period (rub-outs).
2. Turn key switch to ON position.
3. Select mode 'Load from RS' (Section 4.2.2). Code 'b0' will appear in the INST field.
4. Type in required baud rate on digit keyboard (Table 4-9). If 300 baud must be selected, numeral 2 is entered, which will change 'b0' into 'b2'. If 110 baud is needed, a numeral need not be entered and code 'b0' will stay.
5. Depress ENTER key, which will cause code 'P0' to appear in the INST field.
6. Type in required parity on digit keyboard (Table 4-9). If an odd parity is needed, enter numeral 1; this will change code 'P0' into 'P1'. If an even parity is needed, a numeral need not be entered and code 'P0' will stand.
7. Depress ENTER key. Code in INST field will now disappear to indicate that all parameters have been typed in.
8. Depress LINE key and type in address of RAM where loading is to start.
9. Depress SMA key and type in line number of program in peripheral where loading must start (first program word to be loaded), than depress ENTER key.
10. Depress DATA key and select length of program to be loaded by typing in numeral on digit keyboard: numeral 1 for 1k program length to numeral 8 for 8k program length (multiples of 1024 program words). Now depress ENTER key.
11. Start peripheral.
12. Depress START key. 'RUN' LED lights and RS addresses appear in LINE field until address is found where loading is to start (step 9). Now, 'BUSY' LED lights to indicate start of data transmission and abbreviation 'Ld' is displayed in DATA field. Simultaneously, the successive RAM addresses (line numbers) will now appear in LINE field.
13. Dumping ready, 'EOJ' will appear in DATA field and 'BUSY' LED as well as 'RUN' LED will extinguish.

Obviously, loading occurs in the absence of any further command. However, the mode 'Load from RS' can also be used: (1) to verify the program just loaded into the C-MOS RAM, (2) to verify a previously dumped program, Section 4.9. For *verification* to occur, the following step must be interposed:

10a. Depress VFY key then type in numeral 1, which will be displayed in the INST field. Now depress ENTER key.

In step 12, as soon as the RS address appears where verification must commence, 'VFY' will be shown in the DATA field.

When an error is found operation is as explained in the previous section. Also here, the STOP key must be operated before repeating the steps required for dumping (see Section 4.9) or loading (see above).

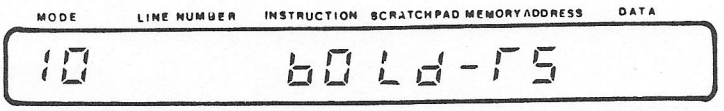
Now, to *re-enter loading,* step 10a should be read as follows:

10a. Depress VFY key, then type in numeral 0, which will be shown in the INST field. Now depress ENTER key.

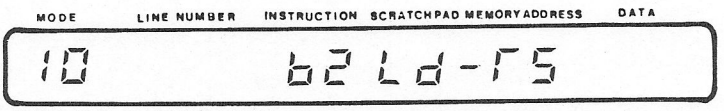
The following example shows how the program loaded according to Fig. 4-28 is verified. That is, the start line number is 300 in RAM and 150 on RS. Required baud rate is 300 baud and parity check is odd.

Example 4-27 Verification of loaded program.

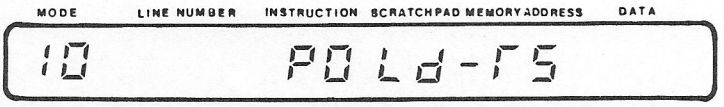
Mode 'Load from RS' selected
(see Section 4.2.2)



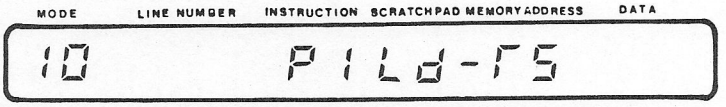
Select 300 baud:
[2]



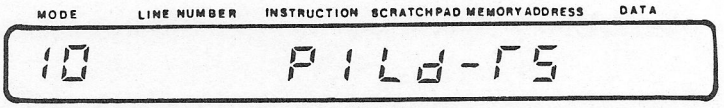
Depress ENTER key:
[ENTER]



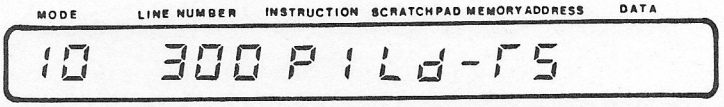
Select odd parity checking:
[1]



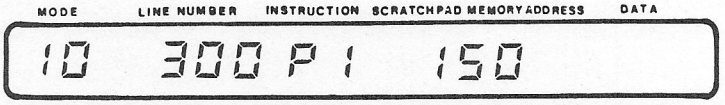
Depress ENTER key:
[ENTER]



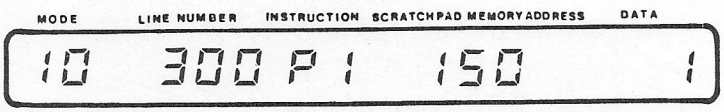
Select RAM start line no.:
[LINE] [3] [0] [0] [ENTER]



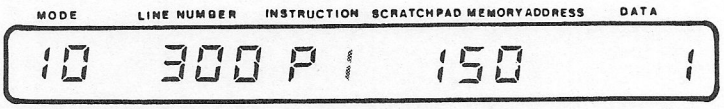
Select RS start address:
[SMA] [1] [5] [0] [ENTER]



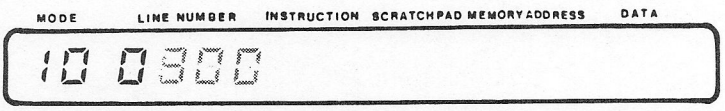
Select length of program to be loaded:
[DATA] [1] [ENTER]



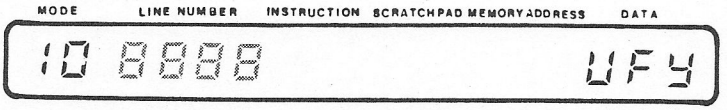
Select verify mode:
[VFY] [1] [ENTER]



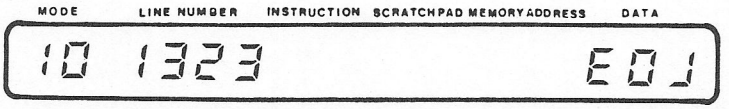
Start peripheral then depress START key:
[START]
(RS addresses shown)



Verification starts
(RAM addresses shown)



End of verification
(display shows last RAM address)



4.13 Specifications

4.13.1 PU20 programming unit

Power supply — A.C. Mains voltage 110 V, 130 V, 220 V or 240 V, selected using voltage adaptor, which is set for 220 V at factory. Three-core flex with earth lead, 2,5 m long.

Connection to PU21 interface — Cable 2 m long consisting of four wire pairs (seven wires used).

Connection to RS peripherals — RS449/423 connector provided at rear of PU20 for connecting a puncher, printer, tape reader, VDU terminal or higher-intelligence equipment. For connections see Fig. 4-29.

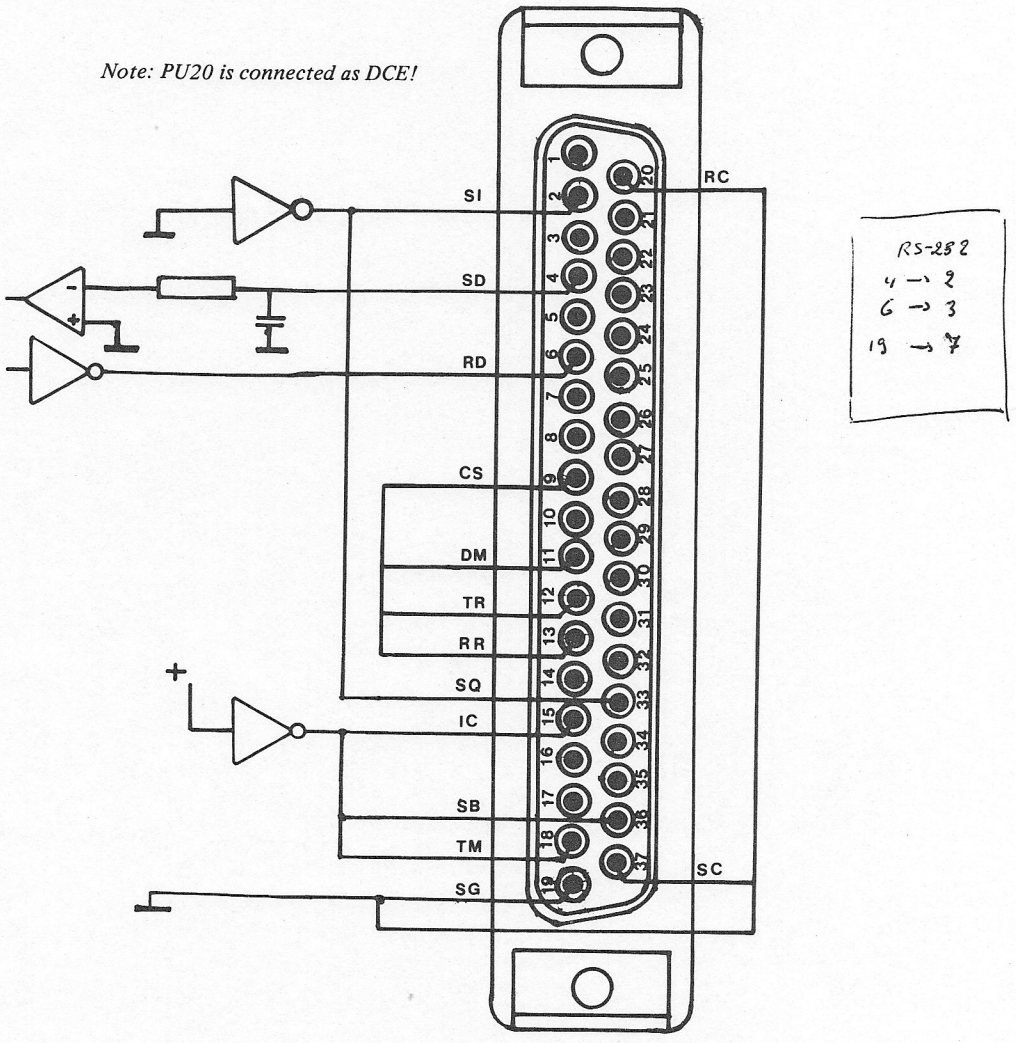


Fig. 4-29 RS449/423 37-pins connector face.

Signal specifications satisfy those laid down in EIA standard RS449/423, data transmission configuration type DT (Data and Timing only). See Table 4-10.

TABLE 4-10 Pinning of RS449/423 connector (37 pins).

pin number	signal name	operational	dummy ON	dummy OFF	jumper 1	jumper 2
2	SI – signalling rate indication		x			
4	SD – send data	x				
6	RD – receive data	x				
9	CS – clear to send					x
11	DM – data mode					x
12	TR – terminal ready					x
13	RR – receiver ready					x
15	IC – incoming call			x		
18	TM – test mode			x		
19	SG – signal ground	x			x	
20	RC – receive common				x	
33	SQ – signal quality		x			
36	SB – standby indicator			x		
37	SC – send common				x	

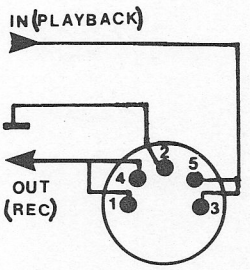


Fig. 4-30 180°, five-pins DIN connector face.

Connection to cassette recorder 180°, five-pins DIN connector available at rear of PU20 for connecting cassette recorder; see Fig. 4-30. For recorder specifications see Section 4.13.2.

Dimensions Width 440 mm; depth 295 mm; height 142 mm.

Weight Approx. 6 kg.

4.13.2 Cassette recorder

Input sensitivity	0,5 mV r.m.s., or better.
Input impedance	$\geqslant 2\,k\Omega$.
Wow and flutter	$\leqslant 0{,}4\%$.
Tape speed	4,76 cm/s ±2% deviation.
Output	0,4 . . . 1,0 V r.m.s. across 20 kΩ.
Output noise and interference	$\leqslant 50$ mV.
Tape	C60 Super Quality FeCr, or better grade.

4.13.3 Storage of program words in EPROMs

For program dumping and loading, EPROMs of either 1k or 2k memory capacity are used, such as the 2758 (1k8) or 2716 (2k8). Each EPROM takes half of a sixteen-bit program word. Fig. 4-31 shows how a program word is stored.

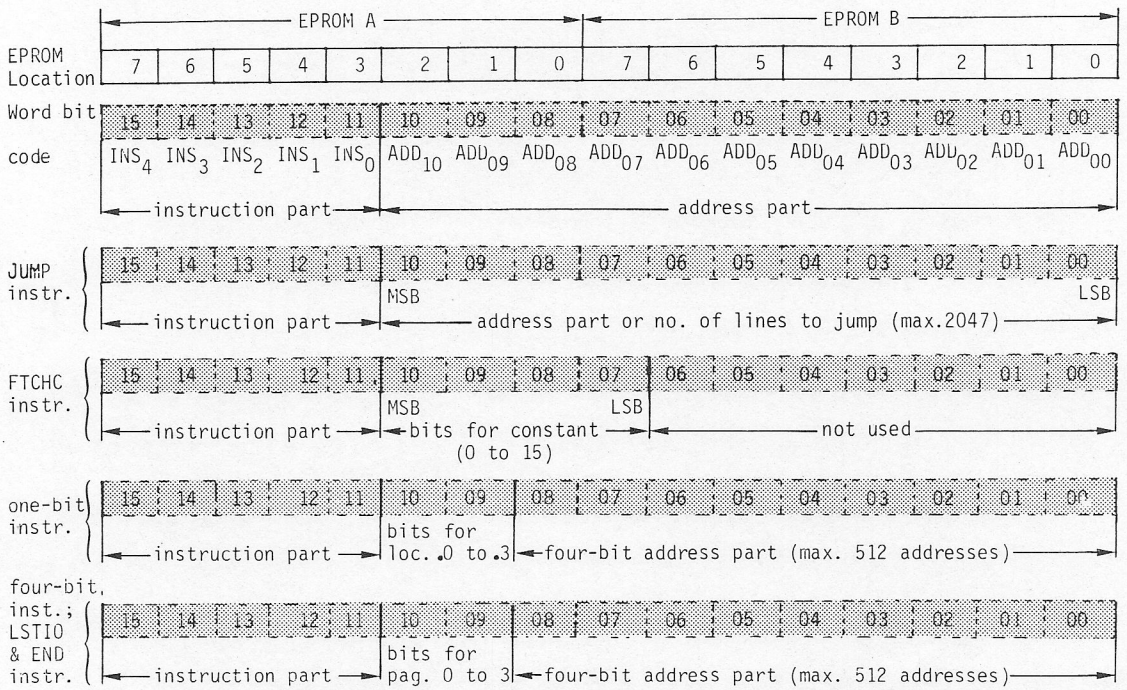


Fig. 4-31 Storage of program word in EPROM.
MSB = most significant bit; LSB = least significant bit.