DATA LINK TEST AND MAINTENANCE PROGRAMS

# INDEX

# REFERENCES

Flow charts

## 1. INTRODUCTION

A program is described which has been used during the development stages of the data links between the SDS 920 and the IBM 1800 on one side and the CDC 6600 on the other side. The same program will remain in use for the maintenance of the links and for this reason a report is given as detailed as possible of its main frame and of its operative procedures.

A test program for a data link is perhaps more essential part of the equipment than a test program for any other machine connected on line, in the sense that nearly all the information is generated by program and the purpose of the test is its transfer from one computer to another.

In order to be useful the following features should be present in it :

a) It should be framed having in mind the actual functioning of the electronics in their detailed sequence of operation rather than their overall functional characteristics.

b) It should be selective in the sense that parts of the equipment are tested separately from others.

c) It should be repetitive in order to facilitate observation of signals on the oscilloscope.

d) It should be capable of recovering from eventual misbehaviour of the equipment, caused by adjustment in course or by error.

e) It should give sufficient evidence of errors, but only on demand, otherwise a continous and known error condition prevents the test from running at its maximum rate.

The data link test consists in effect of three program packages, one for each of the computers mentioned above.

Two operators at least are needed in order to execute a test. One operator is at the 920 or at the 1800 end, he has the choice of one out of three different types of test :

a) Input flag test, which exercises the ability of the link to transfer status information to the 6600.

b) Output flag test, which concerns transfer of status in the opposite direction.

c) Block transfer test, which consists in the generation of blocks of data in the SDS 920 or in the IBM 1800, in the transfer of these blocks to the 6600 and in their transfer back to the original computer, where the pattern received is compared against the pattern generated.

Another operator at the 6600 end has the possibility of selecting the rate at which the test is repeated. Both operators receive typed evidence in the occurrence of errors.

## 2. CDC 6600 Program

It consists of two parts : a central processor main program (C.P. main) of about 1600 octal words and a peripheral processor (PP) program of about 2200 octal words.

The CP main performs the following activities :

a) Conversation with the operator via a teletype unit using two library subroutines TEL2ZA and TEL3ZA.

b) Request for a peripheral processor in order to load into it the PP program, at a rate demanded by the operator of the teletype.

c) Exchange of messages with the PP program in the course of a test sequence.

d) Book-keeping.

e) Conversion from binary to BCD and format preparation for teletype output.

The PP program activities are the following :

a) Exchange of messages with the CP main.

b) Book-keeping.

c) Request for and release of channel 11 (onto which the data link control is connected).

d) Request for and release of channel 3, to which the 6600 Multiplexer is attached. (The data link control uses one of the MUX sub-channels for status input.)

e) Execution of a test as demanded by the computer at the other end of the link.

f) Release of the peripheral processor.

### 2.1 CP Main program

The program starts by typing out the title on the teletype unit :

DATA LINK TEST ****

TYPE ONE OF THE FOLLOWING COMMANDS =

DO, PRINT, GO, STOP

The operator can answer by typing :

a) DO followed by number 1, 2 or 3. It may be recalled that entering of messages via the teletype under the present MUX package system necessitates an initial A+ Control and a final C+ Control.

DO1 means : test link n$^o$ 1 e.g. the SDS 920 link.

DO2 means : test link n$^o$ 2 e.g. the IBM 1800 link.

DO3 means : input status via the MUX subchannel.

The computer will then ask :

TIMES=

The operator can now enter in octal form the number of times he wishes the test to be executed within one PP request.
This number can have any value between 1 and 3777.

The computer asks then :

BY=

The operator enters at this point the number of times that he expects the request for and subsequent release of a PP to be repeated. This number can have any value between 1 and 37777777.
The computer asks next :

ERROR DISPLAY=

To which it can be answered ALL - meaning that all error indications are to be immediately printed, or END - only a summary of errors is printed when the PP is released, or NONE - no error indication is printed. At this point the computer pauses waiting for the operator to either restart the sequence from DO or to type GO.

GO - causes the computer to perform the test as indicated by the commands mentioned above until the counts specified by TIMES and BY have reduced to zero. Typing again GO causes another identical sequence to start. It should be noted that a large value of TIMES causes channel 11 to be reserved for a long time for data link use, preventing other users of the 6600 from using the magnetic tape units located on channel 11. On the other hand a low value of TIMES can make difficult the observation of signals by means of an oscilloscope when users of the tapes delay regularly data link requests for the channel. On the other hand if a catastrophic error is feared it is best to try prudentially TIMES=1, BY=1.

Going back to the title of the program the operator can as a different choice type :

PRINTRES - this causes the first 10 PP words of a buffer called RES to be printed on a line. The first RES location contains the last status word of the link being selected and tested. Following words of RES contain zeros except when an input flag test has been executed in which case they contain a natural progression of numbers in octal form from 1 to 177 in the case of the 920 and from 1 to 77 in the case of the 1800.

PRINTMIP - this causes the first 10 PP words of a buffer called MIP to be printed on a line. This buffer contains the last block of 100 PP words which has been received via the link during a block transfer test. Typing of a second, third etc. GO causes the subsequent locations of the buffers to be printed on the next line. Evidently in order to pass from one sequence PRINT to a sequence DO or vice versa the operator restarts with one of those two commands.

STOP - causes the program to come to an end in which case a last message is typed out : THE END.

Typing of erroneous messages causes the computer to answer : UNKNOWN MESSAGE IN and the title to be repeated.

During a DO sequence the CP main prepares and assembles a word of command for the PP program (EPCO), which contains the following information :

a) Which link is to be used.

b) IS the status to be read via the Multiplexer.

c) How many times the test is repeated before releasing the PP.

d) Are the errors to be printed.

e) Should the PP be released.

If subsequently a GO command is typed in, it requests by means of a TPPW macro instruction a PP into which to load the PP program (LNK). When this request is satisfied the program proceeds up to a WAIW macro. The control word of this macro instruction (PECECO) is set by the PP program. Another control word (CEPECO) is set by the CP main to signify to the PP program that it can proceed analyzing the command word EPCO in order to perform the activities indicated there.

The CP main will come out of the WAIW macro whenever PECECO signals one of the following conditions :

a) Errors

b) End of test sequence (TIMES has been counted down to zero)

c) PP release.

Error conditions are indicated only if the ERROR DISPLAY command
was ALL except for the "link non operational" condition which is
indicated in any case. In most error conditions the test
continues after printing the mnemonic error code with the
exception of "link non operational" and "no disconnect after
function" which cause immediate termination of the test.
At the end of a test sequence the CP main sets EPCO to a value
signifying "release the PP".

The PP program takes the necessary steps and then warns the
CP main that the PP is released. If the error display command
was END a summary of errors is printed out at this stage.
For this purpose a buffer LOCER containing the total times
each error condition was detected is transferred to central
memory from PP memory previously to the release of the PP.
At the same time are transferred also the buffers RES and MIP
mentioned earlier. The CP main checks whether the count
indicated by the value of BY has been reduced to zero. If this
count is greater than zero, one is subtracted from it and the
CP main puts up another request for a PP.

When the count is reduced to zero the complete series of tests
is finished and the program is ready to accept new commands
from the operator.

## 2.2 PP Program (NLK)

The WAIW macro of the CP main is replaced in this program
by a waiting loop during which the CEPECO word set by the CP
main is observed. If no command is given, at the end of the
waiting loop, the peripheral processor is released and this
prevents loss of one PP to the system in case of failure of the
CP main.

If a command is given the PP program analyses the content
of EPCO, mentioned earlier. If the original operators command
was DOI the program goes to a section (GO1) which modifies
several instruction linkages in order to adapt itself to a test
of link I (e.g. 920 link). Having done so, it jumps to GO2,
where it requests channel 11. As soon as the channel is
obtained it sets another waiting loop during which the status
of the link is regularly read in. If after few hundred
milliseconds there is no status indication that a demand or an
error information is advanced by the computer at the other end
of the link, the channel is released and then requested again
300 µsec later.

This gives a chance to tape users on channel 11 to work in
parallel with the link. The process of requesting and releasing
the channel is repeated 6777 times. If at the end there is still
no indication that a test is demanded or that an error is found,
the status request search is abandoned and an error message
is set for the CP main informing it that the other computer is
not ready. If during the status request search the status
indicates that the link is not operational, the search is
terminated, a different message is sent to the CP main, which,
as we know, will command the release of the PP.

67/749/3/mg

| 1) | NO | Data link non operational | – The voltage control mechanism has tripped over due to excessive voltage variation, fall of one voltage line etc. |
|---|---|---|---|
| 2) | ND | No disconnect after a function instruction | – The data link control connection to the DCA is not made, the DLC is switched off, the DL has been non operational for longer than 3 min. |
| 3) | NF | No full after a status request | – A continuous signal on the 920 SYS line or on the 1800 write pulse (channel 01) line prevents the setting of the full Flip-flop. |
| 4) | NSI | WD count $\neq$ 0 after a status input | – This in principle can only be caused by 6600 misbehaviour. |
| 5) | IPE | Input parity error | – Faulty parity circuit, faulty flip-flop in the conversion register, faulty transmission of one bit including the parity bit. |
| 6) | NWE | Number of words error during block transfer | –– Any malfunctioning which causes a disconnect to the 6600 before the end of the block, maybe faulty time limiters. |
| 7) | NR | Non ready | –– the 920 or the 1800 have come to a halt, their programs are destroyed or are not running as expected. |
| 8) | OS | Out of sequence (an unknown message is received from the other computer) | – Faulty input flag circuit, faulty transmission of one bit of status to the DL control or to the 6600. |
| 9) | OPE | Output parity error | – See input parity error. |
| 10) | BTE | Block transfer error detected when comparing the pattern transmitted and the one received | – Link transmission error, conversion register fault, wrong transfer from and to one of the computers. Printing of blocks at both ends shows in which direction the error occurred. |
| 11) | NSR | No status response; after initiating an input flag test no further status is forthcoming. | – The interrupt cables are not connected, a spurious interrupt on a different level causes the computer to be derouted from his test sequence. |
| 12) | IFE | Input flag error | – Wrong transmission, faulty input flag register. |
| 13) | NIW | Non waiting for input ⎫ The expected | – Interrupt missing, 1800 or 920 sequence interruption. |
| 14) | NOR | Non ready for output ⎬ sequence of | – As above |
| 15) | NOW | Non waiting for output ⎭ events is not found | – As above |
| 16) | OFE | Output flag error (1800 only) | – Output flag register error, status line to the 1800 disconnected. |

Table 1

## TABLE 2

| DEMANDED ACTION OR INFORMATION | 920 STATUS CODE (octal) | 1800 STATUS CODE (octal) |
|---|---|---|
| - Block transfer test demand | 2 | 2 |
| - Output flag test demand | / | 32 |
| - Output flag error | / | 52 |
| - Block transfer error | 40 | 72 |
| - Input flag test demand | 177 | 77 |
| - Output parity error | 77 | / |

Demands for a test will cause the program to go to one of the different sections described below.

Input flag test - The three interrupts which are specific to the data links are used to control the test. Since the interrupt level assignments are susceptible to changes, the three interrupts are referred here as OF1, OF2, OF3. OF2 causes the test to start and then to proceed. As soon as the 1800 or the 920 receive this interrupt they send a status word with value equal to 1. The PP program detects this new status, stores it in RES and sends again OF2, which causes the input status word to be increased to 2 and then to 3, to 4 etc.. The test is finished when 177 words have been received from the 920 or 77 words from the 1800 and stored in the buffer RES. The PP program then checks these words one by one and gives OFI if the check is satisfactory or OF3 if an error is detected (IFE).

Output flag test - This test is used only for the 1800 link. The third interrupt line of this link corresponds to 4 bits, OF7 to OFIO, the interrupt being generated whenever any of these bits changes from logical zero to logical one level. For the test the four bits are considered as a word whose value is increased at regular intervals from 1 to 15. After every change the word is reset to zero so that the 1800 can interpret the next change as a new interrupt. After checking the 15 words received, the 1800 sends a particular status word (see table 2) if an error is found.

Block transfer test - This test is closer than the others to the actual operation of production programs. The status demand itself (code equal to 2) corresponds to the input flag code which has the conventional meaning of "920 or 1800 ready for input". To this the PP program answer by setting OF2 (meaning "6600 requests input") the next status expected is "1800 or 920 waiting for input". At this point 100 PP words (12 bits) are transferred via the link to the buffer MIP mentioned earlier. The PP program waits then for the other computer to become "ready for output" and following this status information sets OF1 ("6600 requests output") which causes the other machine to indicate that it is "waiting for output". The next step consists in sending back via the link the 100 words of MIP. When this operation is completed a check is made to find out whether the complete block was received by the 6600. If a number of words error is detected the third interrupt is sent to the other computer and at the same time a particular output flag is set to indicate this condition (OF7 for the 1800, OF8 for the 920). In the case of the 920 only, the

status read at the end of the input block transfer indicates whether a parity error was detected. If this happened the third interrupt again is sent to the 920 but with a different output flag e.g. OF8.

Reading the status via the Multiplexer - This test can be performed without special demand from the other computer. It is in effect the 6600 operator who by typing DO3 causes EPCO to contain a command for the peripheral processor to execute this test. The test begins by requesting channel 3, inputting 1 word via the multiplexer subchannel connected to the link control and releasing the channel. The input word contains four bits of 920 status and four bits of 1800 status. This word is stored at the first location of buffer RES and it can be printed out as explained earlier.

3. **1800 Program**

The data link test operator being also in charge of the complete operation of the 1800 few details are given about the use of this computer. Before reading the program in the following conditions must be satisfied :

a) Computer, data link terminal, typewriter, air conditioning are switched on.

b) Mode switch is on run, the switches in the two bottom rows (sense, program, data entry switches, etc.) are all down.

c) The subroutine library tape is loaded and the deck is in the "ready" state.

The 1800 is reset by pressing once immediate stop and then reset keys. The program is read in by placing the cards in the reader and pressing start button on the reader command panel followed by depression of program load button on the top row of buttons of the computer console.

When the program has been read in the run out button at the reader must be pressed in order to extract the last two cards of the program from this machine. The program is now loaded and waits at location 2000. The operator presses the start button on the 1800 console, which causes the program to run and to light up the typewriter select indicator. Six numbers from 1 to 6 must be entered at this point. They represent a choice of a sequence of six tests and their significance is the following :

1 - Block transfer test, pattern alternate 1,

    even words contain -    1111111111111111

    odd words contain -    0000000000000000

2 - Block transfer test, pattern alternate 2,

    even words contain -    1010101010101010

    odd words contain -    0101010101010101

3 - Block transfer test, pattern alternate 3,

    even words contain -    1111111100000000

    odd words contain  -    0000000011111111

4 - Block transfer test, pseudo random pattern

       This pattern is generated as follows. Part of the program itself between an initial and a final address is taken as a convenient source of heterogeneous words which are manipulated even further in order to produce a more random pattern. Initially a variable SOM is filled with the number 3777B and a location counter DEB1 is set to the initial address. The content of the location with address DEB1 is shifted one position to the right and then subtracted from SOM. This result is stored in SOM and in a location of the random pattern buffer. DEB1 is increased by one and two tests are done : 1) If SOM $<$ 0 this location is filled again with 3777B. 2) If the final address is reached, DEB1 is set again to the initial address.

       The following words are generated in the same way. When the buffer is completely full SOM and DEB4 are saved as the initial parameters of the next random pattern.

5 - Input flag test

6 - Output flag test

       The six numbers can be entered in any wanted order giving a total of $6^6$ possible different sequences. Having entered these values the computer goes to a long waiting loop (ATT), from and to which will regularly return in the course of the test. The sense switches 0, 1, 2 on the console control the program in the following way :

Sense switch 0 - causes printing out of a line of pattern in case of error being detected. Two octal bits on the left hand side of the page represent the word number. The following 16 binary digits represent the bits of the word received from the 6600. The two words are not identical and visual inspection detects which bits are wrong.

       The same switch causes also in case of error printing of the output flag buffer (SOF). The output flag values should range from 1 to 15 as explained in section 2.2.

Sense switch 1 - causes the block of words transmitted and the one received to be printed out independently from the eventual presence of errors. The position of both switches is observed at the end of each line, in this way further printing can be prevented by returning the switch to its off position before the end of the line.

Sense switch 2 - causes the typewriter to be selected again in order to enter another sequence of six numbers corresponding to the tests mentioned above but only at the end of the current sequence of tests. The program performs the following functions :

a)  Execution of tests

b)  Selection of tests

c)  Check of patterns

d)  Input from typewriter and associated code conversions

e)  Output to typewriter and associated code conversions

At the beginning of the long waiting loop ATT the computer sends out an input flag code corresponding to a demand for the first test of the selected list. Table 2 shows these demand codes. If at the end of the waiting loop, no interrupt (OF1, OF2, or OF3) has arrived via the link, the input flag demand is reset to zero. The 1800 waits just a little longer to take care of an eventual interrupt pulse being already on its way and then terminates that test by printing NR (6600 non ready).

If an interrupt arrives meaning that the 6600 has recognized the demand, the test is performed and then the program analyses the next number on the list and puts out a demand for corresponding test before entering again the waiting loop ATT. Everything proceeds in the same way up to the last number on the list. At that point a check is made to find out whether the sense switch 2 is up. If it is not up the same list is used again to start another series of demands. The actual logic of the tests and the relative exchange of status between the two computers have been described in section 2.2, it will be mentioned here that three parts exist around the waiting loop ATT, one part corresponds to the block transfer test (CDCIN), another to the input flag test (TIF), another to the output flag test (TOF).

The three interrupts which control the tests (OF1, OF2, OF3) will direct the program to jump to a different part of CDCIN, TIF or TOF according to the following table.

Table 3

|  | OF1 | OF2 | OF3 |
|---|---|---|---|
| Block transfer test | Output (6600) request | Input (6600) request | Number of words error |
| Input flag test | Test completed and O.K. | Increase flag value | Test completed error found |
| Output flag test | Unexpected | Unexpected | Store associated PISW (four bits) |

During a block transfer test having received interrupt OF2 and transferred a block to the 6600, the program returns to ATT. After OF2 the block is transferred in the opposite direction and there is no return to ATT. A short wait is set up instead, at the end of which if no OF3 arrives the program carries on checking the pattern received. If, on the other hand, OF3 arrives NWE is typed out. The mnemonic codes given here have the same meaning as the ones in table 1. During an input flag test

OF2 causes the program to send a new input flag with a value increased by 1 and to return to ATT. OF1 is received if the test ends correctly and OF3 if the 6600 detects an error in which case IFE is typed out.

During an output flag test the process interrupt status word (PISW) of the interrupt level corresponding to OF3 (this is the status contained in the output flags OF7 to OF10) is stored in a buffer named SOF and a check is made to find out whether it has the correct value and whether the total expected number of words have been received. If one value is wrong a program marker is set which at the end of the test will cause printing of the mnemonic code OFE. Interrupts OF1 and OF2 are not expected in this case, and therefore if they are received the OFE marker is set again and in addition the first word and the second word respectively of SOF contain four ones in the positions corresponding to the PISW of the interrupt level OF3.

4. **920 Program**

As for the 1800 program the data link test operator must be able to operate some of the controls of the computer in order to load the test program in the 920. Before attempting to read the program in the following conditions should be satisfied :

a) Computer, data link terminal, interface unit, typewriter and ventilation system are switched on.

b) Register switch is on position C.

The 920 is reset by putting the run key on idle and pressing the start button. In order to load the program, the paper tape is placed in the reader (there should be a binary boothstrap part before the actual data link test). The run key is set on position run and the fill key is raised once. The binary boothstrap goes through and then the tape stops. In order to proceed the run key is lowered to its idle position and is set on run again. The whole length of tape goes through the reader and then the computer stops. In order to initiate the program it is necessary to execute manually a branch to its entry point, which is at location 1300 octal. For this purpose the run key is lowered to idle, the register switch remains on C and by means of the bottom row of buttons the following octal word : 00101300 is entered in the C register of the machine and can be read on the register display indicators. The run key is then raised to its run position and the program starts.

The program is controlled by means of the breakpoint switches 1, 2 and 3. Breakpoint 1 causes the typewriter to be selected in order to enter a sequence of four numbers from 0 to 3, corresponding to the four following tests.

0 - Block transfer test, pattern alternate 1,

    even words contain -      1111111111111111111111

    odd words contain -       0000000000000000000000

1 - Block transfer test, pattern alternate 2,

    even words contain -      1010101010101010101010

    odd words contain -       0101010101010101010101

2 - Block transfer test, pseudo random pattern,

The method of generation of the pattern is similar to the one utilized
for the 1800 (see section 3) but with the following differences.
SOM is initially set equal to zero. A maximum value of SOM is defined
and whenever by the addition of instructions to SOM this'number
becomes larger than its maximum, it is set again to zero.

3 - Input flag test,

The four numbers can be entered in'any wanted order giving a total
of $2^4$ possible different sequences.

Breakpoint 3 causes printing of words of the pattern at the'
end of a block transfer test, if it is found that they contain errors.
The first two octal figures on the left hand side of the page represent
the number of the word in the block. The following eight octal figures
correspond to the word transmitted to the 6600 and the next eight to the
word received from the'6600. Visual comparison of the two words detects
the cause of the error. Breakpoint 2 causes the block of words transmitted
and the one received to be printed out independently from the eventual
presence of errors. The logic of the program is very similar to the one
of the 1800 program, the use of the three interrupts OF1, OF2, OF3 is
identical. Here however the output flag test is not included because no
information equivalent to the PISW of the 1800 can be entered into the
920.

In the case of the block transfer reception of OF3 can have the
meaning of input parity error in addition to the other of number of words
error. The input parity error is detected by the data link control during
a block transfer to the 6600 and is read by means of a subsequent status
input. But it is only at the end of the transfer back to the 920 that the
6600 program sends interrupt OF3. As explained in section 2, OF3 is
accompanied by either or both OF6 and OF7 to indicate respectively the
detection of a number of words error and of an input parity error. The
920 program investigates the presence of the signals OF6 and OF7 by means
of SKS instructions (skip if signal is not set).

With the exemption mentioned above table 3 is also valid in this
case. Another slight difference between the two programs is the lack in
this case of an unique waiting loop (ATT of the 1800 program) which is
replaced by individual waiting loops for each of the two possible types
of test, block transfer and'input flag test.' The entries to those two
tests have the same names e.g. CDCIN and TIF.

In addition to the word by word comparison performed on the
pattern received from the 6600, the 920 executes an hardware parity check.
This check is not done at the moment the data enter the computer, but
as soon as the memory is unloaded. A parity error detected in this way
can therefore be due to other reasons besides wrong transmission over the
link. A parity error detection causes the 920 to halt with the memory'
parity indicator on, if the corresponding key is on its halt position.

References

1. Control Data 6600 Series Computer Systems Reference Manual.

2. IBM 1800 Data Acquisition and Control System Functional Characteristics.

3. IBM Special Feature Report No. 2-0016.
   High Speed Digital Input and Output.

4. SDS 920 Computer Reference Manual SDS 900009B.

5. F. Marciano and H.J. Slettenhaar - A data link from PS experimental area to Mercury computer CERN report 65-5.

6. F. Marciano - A data link between two computer CERN 65-40.

7. S. Bunodière, J. Austin - Manuel du Data Link CERN DD/DA/66/16.

8. F. Marciano, IBM 1800 - CDC 6600 Data Link
   Preliminary specifications DD/DA/67/5.

# DATA LINK IBM 1800 TEST PROGRAM

1

**WAITT**

SWiTCH 3 ON — NO / YES

LOAD 6 NUMBERS IN SELB BUFFER WITH TYPEWRITER

NBPAT = 1

TRANSFORM SELB FROM TYPEWRITER CODE TO INTEGER

STORE IN SEL

**DEBUT**

FILL BUFFERS: LOC 1, LOC 2, LOC 3, WITH PATTERNS 1, 2, 3,

**CHECK**

50 IN INDEX 1

LOCER = 0

FORM: ADRER+(INDEX 1) PATIN +(INDEX 1) - PATOU +(INDEX 1)

ADRER (INDEX 1) = 0 — YES

LOCER = 1

INDEX 1 = INDEX 1-1

INDEX 1 = 0 — PRI YES / NO

CHOOS — NBPAT = NBPAT +1

NBPAT > MAXPT — YES / NO

LOAD SEL (NBPAT) STORE IT IN ST1

PUT ST1 IN INDEX 1

INDEX = INDEX 1-1

PATTERN VARIABLE

INDEX 1 = 0 — YES / NO

INDEX 1 = INDEX 1-1 — NO

INDEX 1 = 0 — YES

INDEX 1 = INDEX 1-1 — NO

INDEX 1 = 0 — YES

INDEX 1 = INDEX 1-1 — NO

INDEX 1 = 0 — YES

INDEX 1 = INDEX 1-1

INDEX 1 = 0 — YES / NO TIF / TOF

INDEX 2 = 50

INDEX 2 = 50

INDEX 2 = 50

LOAD LOC 3 +(INDEX 2) STORE IN PATIN +(INDEX 2) INDEX 2= INDEX 2-1

LOAD LOC 2+(INDEX 2) STORE IN PATIN +(INDEX 2) INDEX 2= INDEX 2 -1

LOCD LOC 1 INDEX 50 STORE IN PATIN+(INDEX 2) INDEX 2 =INDEX 2-1

INDEX 2 ≮ 0 — NO / YES

INDEX 2 ≮ 0 — NO / YES

INDEX 2 ≮ 0 — NO / YES

CDCIN

CD CIN

C DCIN

## VARIABLE PATTERN

```
PATV IN INDEX 1
```
↓
```
50 IN INDEX 2
```
↓
```
LOAD DEB 1+ INDEX 1
SHIFT 1 PLACE RIGHT
STORE IN PA
```
↓
```
LOAD SOM SUBSTRACT
FROM PA STORE IN
LOC 4 – IND 2
```
↓
◇ SOM ≤ 0 — YES → `SOM = MK3`
↓ NO
```
STORE A REG. IN SOM
```
↓
```
SUBSTRACT 1 FROM IND 1
```
↓
◇ INDEX 1 < 0 — NO →
↓ YES
```
INDEX 1 = 101
```
↓
```
INDEX 2 = INDEX 2 - 1
```
↓
◇ INDEX 2 ≤ 0 — NO →
↓ YES
```
STORE INDEX 1 IN PATV.
```
↓
```
INDEX 2 = 50
```
↓
```
LOAD LOC 4 + INDEX 2
STORE IN PATIN +(2)
INDEX 2 = INDEX 2 - 1
```
↓
◇ INDEX 2 ≤ 0 — YES →
↓ NO

### CDC IN

```
16 IN INDEX 2
LOAD PATOU + 52 +(1)
IN A REGISTER
```
↓
```
STORE A REG. IN STR 2
```
↓
```
KEEP THE LAST RIGHT
BIT OF STR 2 REPLACE
IT BY ITS IBM TYPE –
WRITER CODE STORE
IN BUF + 20 +(2)
```
↓
```
STR 2 IN A REGISTR.
SHIFT 1 PLACE
ON RIGHT
```
↓
```
INDEX 2 = INDEX 2 - 1
```
↓
◇ INDEX 2 > 0 — YES ← / NO →

## PR 1

```
– 50 IN INDEX 2
```
↓
```
CARRIAGE RETURN
IN BUFF + 1 N° OF
WORDS TO PRINT PER
LINE IN BUFF
```
↓
```
WAITT
```
↓
◇ SWITCH 1 ON — NO →
↓ YES
◇ ADRERR + 52 +(1) = 0 — YES →
↓ NO
```
TRANSLATE LINE N° IN IBM
TYPEWRITER CODE
```
↓
```
STORE IT IN BUFF + 2
2 SPACES IN BUFF + 3
```
↓
```
16 IN INDEX 2; PATIN +52+(2)
IN A REGISTER
```
↓
```
STORE IN STR2
```
↓
```
KEEP THE LAST RIGHT BIT
OF STR2 REPLACE IT BY ITS
VALUE IN TYPE WRITER CODE
STORE IN BUFF +3 +(2)
```
↓
```
STR2 IN A REGITER; SHIFT
1 PLACE ON RIGHT.
```
↓
```
INDEX 2 = INDEX 2-1
```
↓
◇ INDEX 2 > 0 — YES / NO →
↓
```
PRINT BUFFER
BUFF
```
↓
```
INDEX 1 = INDEX 1 + 1
```
↓
◇ INDEX 1 ≤ 0 — NO →
↓ YES
```
NBRR = 15 WAITT
```
↓

## PR 11

PR 11

```
┌─────────────────────────┐
│   50  IN  INDEX  1       │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│  CARRIAGE  RETURN  IN    │
│  BUFF + 1 AND N° OF WORD │
│  TO PRINT PER LINE       │
│  IN  BUFF.               │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│        WAITT            │◄────────────────────────────────┐
└─────────────────────────┘                                 │
            │                                                │
            ▼                                                │
      ╱───────────╲      NO                                  │
CHOOS◄── SWITCH  2  ON                                       │
      ╲───────────╱                                          │
            │ YES                                            │
            ▼                                                │
┌─────────────────────────┐                                 │
│  TRANSLATE  LINE N° IN   │                                 │
│  IBM  TYPEWRITER         │                                 │
│  CODE                    │                                 │
└─────────────────────────┘                                 │
            │                                                │
            ▼                                                │
┌─────────────────────────┐                                 │
│  STORE  IT IN BUFF. + 2  │                                 │
│  AND  2  SPACES  IN      │                                 │
│  BUFF. + 3               │                                 │
└─────────────────────────┘                                 │
            │                                                │
            ▼                                                │
┌─────────────────────────┐                                 │
│  16 IN INDEX 2 PATIN     │                                 │
│  + 52 + INDEX 1          │                                 │
│  IN  A  REGISTER         │                                 │
└─────────────────────────┘                                 │
            │                                                │
            ▼                                                │
┌─────────────────────────┐                                 │
│     STORE  IN  STR 2     │◄──────┐                         │
└─────────────────────────┘       │                         │
            │                      │                         │
            ▼                      │                         │
┌─────────────────────────┐       │                         │
│  KEEP LAST RIGHT BIT     │       │                         │
│  OF STR2 REPLACE IT BY   │       │                         │
│  ITS IBM TYPEWRITER      │       │                         │
│  CODE. STORE IN BUFF+3+IND2│     │                         │
└─────────────────────────┘       │                         │
            │                      │                         │
            ▼                      │                         │
┌─────────────────────────┐       │                         │
│  STR2 IN A REGISTER      │       │                         │
│   SHIFT  1  PLACE ON     │       │                         │
│  RIGHT.                  │       │                         │
└─────────────────────────┘       │                         │
            │                      │                         │
            ▼                      │                         │
┌─────────────────────────┐       │                         │
│  INDEX  2 = INDEX 2 + 1  │       │                         │
└─────────────────────────┘       │                         │
            │                      │                         │
            ▼                      │                         │
YES   ╱───────────╲   NO           │                         │
◄─────  INDEX  2 ≥ 0  ─────────────┘ (continues right)       │
      ╲───────────╱                                          │
```

2 SPACES IN BUFF. + 20

16 IN  INDEX 2 PATOU + 52 + INDEX 1 IN A REGISTER

STORE  IN  STR 2

KEEP RIGHT BIT OF STR2  REPLACE IT BY ITS IBM TYPEWRITER VALUE STORE IN BUFF. + 20 + (INDEX 2 )

STR2  IN A REG.  SHIFT 1 PLACE  ON RIGHT.

INDEX 2 = INDEX  2 − 1

YES  ╱── INDEX ≥ 0 ──╲ NO

PRINT  BUFF.

INDEX 1 = INDEX 1 + 1

╱── INDEX 1 ≤ 0 ──╲  YES

NO

CHOOS

PRINT MESSAGE TO BEGIN THE TEST

RECEIVE 1 WORD IN INME1 BY TELETYPE

INME 1 = STOP → YES → FIN

NO

KEEP LEFT PART OF INME1

THAT PART = PRINT → YES

NO

KEEP 12 LEFT BITS OF INME 1

THAT PART = DO → YES

NO

CONTE = O
KEEP 12 LEFT BITS OF INME 1

THAT PART = GO → NO → PRINT MESSAGE

YES

PRIDO = ... = 1 ... = 0

= 2

PRINT 1 LINE OF BUFMI

CONVERT DO IN BINARY AND STORE IN LEFT PART OF EPCO

SEND NEW MESSAGE AND RECEIVE 1 WORD IN INME 1

SHIFT INME 1 36 LOC ON RIGHT CONVERT TO BINARY

SHIFT BINARY 36 LOC ON LEFT ADD TO EPCO

SEND A NEW MESSAGE RECEIVE 1 MORE WORD IN INME 1

SHIFT 12 BITS ON RIGHT CONVERT IN BINARY STORE IN NOTIM

SEND A NEW MESSAGE RECEIVE 1 WORD IN INME 1

M K P R = 0

INME 1 = PRALL → YES

NO

INME 1 = PREND → YES → O IN X 7 REGISTER

NO

INME1 = PRNONE → YES → MKPR = 1

NO

PRINT 1 LINE OF BUFRE

STATUS = 1
CEPECO = 1
EPCO = KETH

ASK FOR A PP AND BEGIN TO EXECUTE PP PROG LNK

STATUS = 0 → NO

YES

PECECO = 1
CEPECO = 0

WAIW MACRO

PECECO - 1 ... = 0 ... < 0

> 0

PRINT ERROR

NOTIM = NOTIM - 1
PECECO = 0  CEPECO = 1
WAIW

NOTIM = 0 → NO → PRINT OK

YES

MK PR = 0 → NO

YES

PRINT ERRORS

1 IN X7 REGISTER

SHIFT X7 24 PLACES TO LEFT

CLEAR BITS 24-36 OF EPCO

ADD X7 TO EPCO

STORE IN KETH

PRIDO = O

# CDCNR

```
          WAITT

       NBR = NBR + 1
```

YES ← `NBR = MXBRR` → NO

```
STORE CARRIAGE RE                STORE  2  SPACES
TURN AND SPACE                    IN  BUFF + 1
IN BUFF+1   NBR = 0
```

```
       BUFF = 2
       BUFF + 2 = NR

       PRINT BUFFER
       BUFF
```

CHOOS

**NEXT 2**

`OFE = 0` — YES → `IFE = 0` — YES → `NWE = 0` — YES

NO ↓ ... NO ↓ ... NO ↓

```
   WAITT            WAITT            WAITT

   OFE = 0          IFE = 0          NWE = 0
```

`SWITCH 1 ON`

NO → ... YES ↓

```
-15 IN INDEX  2
```

```
PREPARE BUFFER          WAITT          PREPARE BUFFER        PREPARE BUFFER
BUFF TO PRINT                          BUFF                  BUFF
OFE                     INDEX 1 = 4     TO PRINT  IFE        TO  PRINT  NWE

-1 IN INDEX  2
```

```
KEEP LAST RIGHT BIT      PRINT BUFF       PRINT  BUFF
OF SOF + (2) REPLACE
IT BY ITS IBM TYPE-        WAITT            WAITT
WRITER CODE
```
CHOOS ... CHECK

```
STORE IN BUFF + 2 +(1)
SHIFT SOF + (2) ONE
PLACE TO THE RIGHT
```

<u>WAITT</u>

```
INDEX 1 = INDEX 1 - 1      WAIT UNTIL TYPE
                           WRITER IS  IDLE
```

`INDEX 1 < 0` — NO →

YES ↓

```
RETURN JUMP TO THE
CALLING SEQUENCE

PRINT BUFFER
BUFF

ADD 1  TO INDEX 2
```

`INDEX 2 > 0` — YES →

NO ↓

```
      WAITT
```

CHOOS