H. L. Gross

CRC 102-A

NOTES ON FLOW DIAGRAM

The CRC 102-A is a general purpose computer combining a magnetic memory, electronic arithmetic units, and electro-mechanical input-output devices. Computations are made by use of coded programs entered into the main memory in the same manner as the initial data that is to be operated on. There are twenty-five arithmetic, logical, and input-output commands with a mean total time of 60 milliseconds per arithmetic or logical command. Each command is identified by special binary coding which select certain logical blocks of the arithmetic unit.

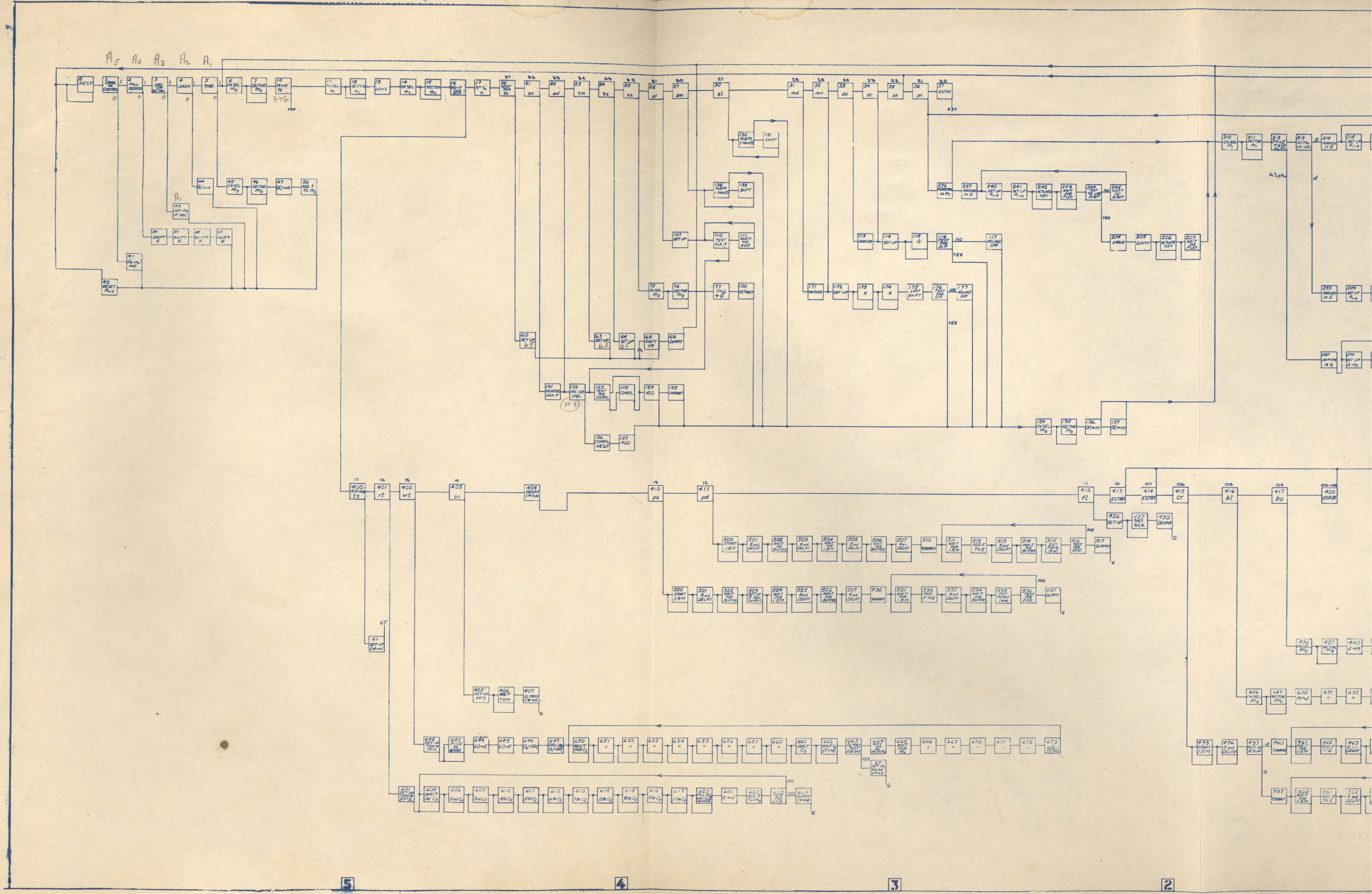
The main memory has a capacity of 1024 words (where a word means a number or a command). Each word is contained in a "cell" each of which is numbered from 0000 to 1777 octally consecutively. There is, in addition, a register of eight cells which is used as a buffer register for input-output devices and for a quick access memory during computation. Each cell consists of a word time of 42 binary bits. For convenience these 42 bits are broken down in the computer basic timing into 14 octal digits. All numbers to be operated on must be in the octal notation and consist of a two digit sign and overflow position at the high order end of the word followed by the number of twelve digits magnitude.* A command consists of a two digit instruction followed by three four-digit addresses of the cells of the numbers to be operated on. These four-digit addresses are titled from left (MSD) to right (LSD) m_1, m_2, and m_3.

Numbers may be entered in the decimal form as well as in octal but it is necessary to convert them by a subroutine during computation to standard binary notation before they may be used in the arithmetic unit of the computer. Decimal numbers consist of a sign digit and nine decimal digits for magnitude. All numbers are considered as binary fractions with the binary point to the left between the sign and the magnitude of the number. With this fixed point convention any operation which results in a number greater than unity will result in an overflow which will stop the computer unless an overflow is anticipated and the computer is instructed to ignore or correct the overflow.

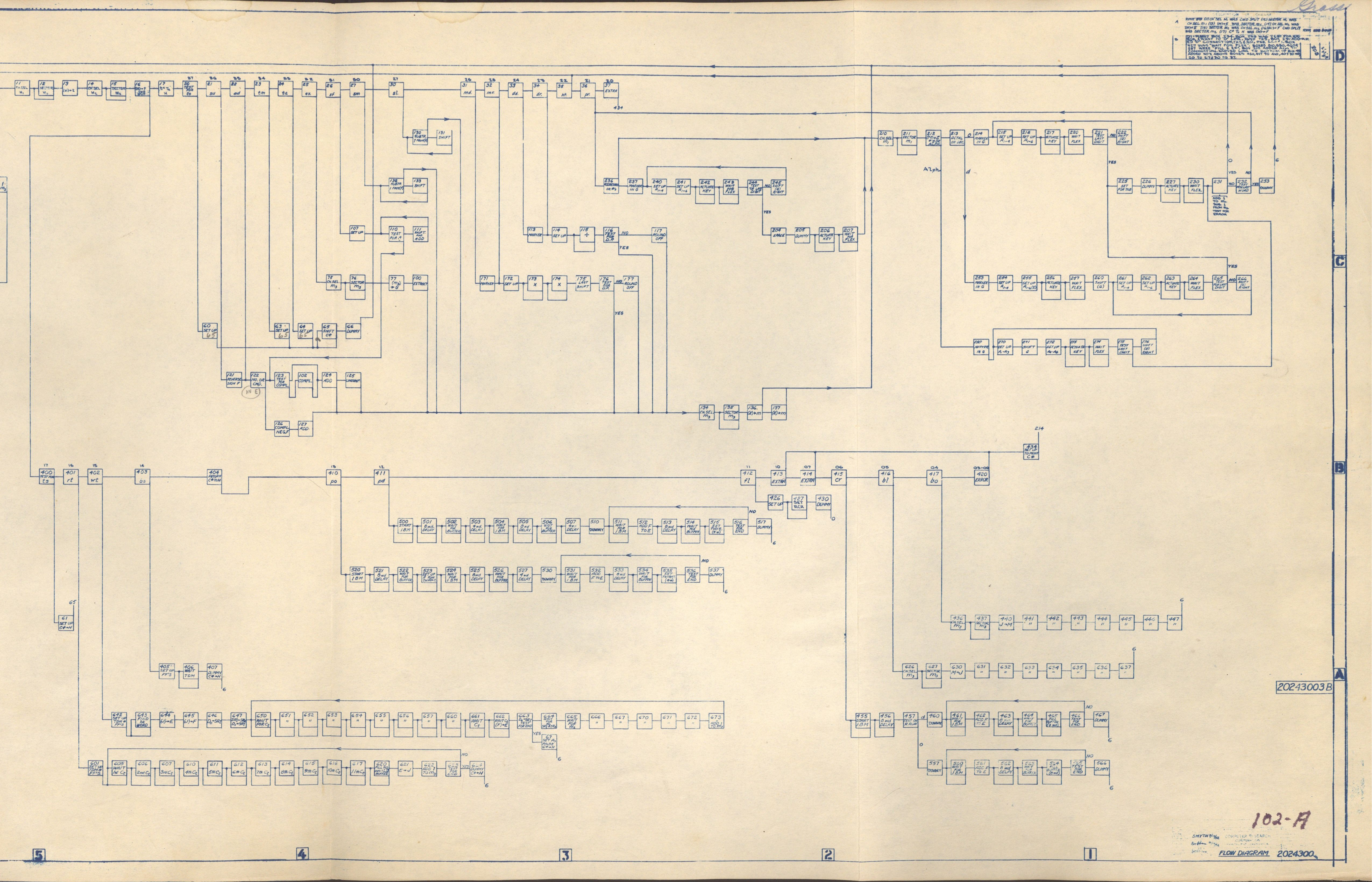
The filling of the initial numbers and instructions may be accomplished manually on a Flexowriter or by the use of Flexowriter paper tape. During compute additional data can be filled from the Flexowriter paper tape unit, magnetic tape units or from an IBM Summary Punch. The contents of any cell on command may be printed out on the Flexowriter, perforate in tape on the Flexowriter, read out into magnetic tapes or punched in IBM Summary Punches.

All operations performed in the 102-A whether fill, arithmetic, logical, or input-output, make use of four one word recirculating memories, E, F, G, and H, ten arithmetic flip-flops A_1 through A_{10}, one control flip-flop, K, and five channel selector flip-flops, L_1 through L_5. All recording on the magnetic memory is controlled by two record flip-flops R_1 and R_2. One eight word buffer register J is provided for input-output or for a quick access memory. Each one word register has associated flip-flops which allow for the logical shift of the binary number either right or left.

* We will use "digit" for octal digit; "bit" for binary digit.



A5 A4 A3 A2 A1
6 REST
2 FILL ZEROS
4 DASH
5 TAB.
401 rt
402 wt
403 bs
410 po
411 pd
412 fl
413 EXTRA
415 cr
416 bl
417 bo
420 ERROR
5
4
3
2



5
4
3
2
1
D
C
B
A
20243003B
102-A
FLOW DIAGRAM 2024300

All addresses are located by means of a permanently recorded address channel M_w on the main memory. Digit and pulse positions are determined by a digit counter and a pulse counter respectively. These are driven by a master clock recorded on the drum and count pulses P_0 through P_2 and octal digits O_0 through O_{13}.

In the design of the logical networks of the 102-A a rather unique system has been developed. In this system a minimum number of logical components has been chosen so that when properly interconnected by the logical diode nets they will carry out the most complex operation which the computer is designed to perform. By properly assigning tasks to these elements they may be made to execute any of the operations required. Since while solving one programmed instruction each of these components must do many varied tasks, they must be controlled in a logical sequence during the time the computation takes place. The controlling device is called the program counter. The program counter determines which of a series of operations the computer is performing at any given time. This program counter is entirely an operational unit of the computer and the operator has no programmed access to it. In the 102-A an arithmetic, logical, or an operational command is broken up into a sequence of many individual operations each of which takes place in a period of time called one-word time (42 bit or pulse times.) During each word time the logical units are used as directed by the configuration of the program counter to perform the particular operation of the sequence.

Using this design method it is then possible to indicate the sequence of operations which are performed automatically by the computer during the execution of any single command by means of a flow chart. Each different use of the logical units is called an operational block. Each block is given the number which correspons to digits of the program counter configuration which calls for the performance of the operations designated in that block.

The sequence of the operations performed for any one given command depends on whether the program counter counts in order from one octal number to the next, or whether it skips to any other nonsequential number. This process of counting or skipping is controlled by the K flip-flop which if false indicates the program counter should add one to its value and in the next word time perform the next operation as indicated by the network logic. If the K flip-flop is true at the end of a word time the program counter is set up to a new octal number as indicated by the logic of the block that it is in, and performs a different operation. The function of each of the operational blocks shown on the flow diagram is described in the operations chart prepared for use with the flow diagram. The operations chart indicates the use of all of the logical components used in each operational block.

In the following explanations of various command and routines used in the 102-A all commands will begin at block 0 and will proceed through each operational block until the execution of the command is completed. The basic use of all of the logical units will be indicated and where necessary explanations will be added. In arithmetic and logical programs the E, F, and G registers contain the numbers to be operated on or the results at the end of the operation. The H register is used to locate the numbers or commands and hold the instructions.

The address of the command to be performed is called the "Control Number". When beginning a computation it is necessary to indicate the address or control number of the first command to be used. The first command can be located in any cell but the following commands should be filled in the immediately following cells in order (except for conditional transfers). This is because the computer, after locating the address of the command given by the control number, adds one to the control number to be ready for the next command. In this manner once the initial address of the first command is entered the computer will proceed with all of the filled commands in sequence until the last instruction has been carried out, at which time it will halt. The computer has the ability to modify its own commands and so it is not always true that commands must be filled in consecutively numbered cells. Non-sequential skipping occurs in the use of Test Magnitude, Test Algebraically and Test for Overflow and Test Switch Commands (See section on programming).

The control number is always entered in m_2 of the one-word register H. This address is then looked up and the command contained in that cell is transferred to H. At the same time a one is added to the control number in m_2 of H, and H is transferred to the one-word register G. The number whose address is now contained in m_1 of H is now found and put into the one-word register E. The number whose address is in m_2 of H is found and put into the one word register F. The word in G is now transferred back from G, to H. This is to clear the G register and make it available for use during computation. The H register now contains the command (operation to be performed), the address of the number in E, the new control number in m_2 and the address of the cell in which the results of the operation are to be placed (or the optional control number in the case of the "Test" commands) in m_3. The command in the two most significant digits of H is then tested to determine the logical operation block in which computation should begin. The command for any operation is in O_{12} O_{13} of the instruction word. The proper block to start computation is found by testing O_{12} O_{13} of H. If H is all 1's, then the correct block has been reached and K is left true for a skip-out. If not, K is set false and one is added to the command number, and is again tested. This continues until the break-away block is reached. The command numbers are coded up so that H will be 01$\overline{111}$1 at the proper block.

A brief description and operation of each operational block for arithmetical and logical programs follows. All operations are carried out by standard binary methods for addition, subtraction, and multiplication. Division is accomplished by a modification of standard techniques. A number (X) with parenthesis around it indicates an operational block. Any capitalized character such as A_2 indicates a flip-flop. m numbers, i. e., m_1 refers to the address. (m_1) means the contents of the cell whose address is m_1.

The first step in any operation is to fill the initial data into the machine. As stated previously, this can be done in several ways but in this case we will consider manual input on the Flexowriter.

If any key on the Flexowriter is stuck, flip-flops A_1 through A_5 are set up to conform to the binary code of the digit entered or a configuration representing the control function to be performed. The computer then tests A_1 - A_5 in each succeeding operational block as indicated in the following explanations to determine which operation is to be performed. To fill in either sequences of commands or numbers it is necessary to enter the address of the first one in octal as follows: Select octal number system by pressing the "O" key. This sets the control flip-flop K false, A_4 and A_5 true, and allows the program counter to count out of (0) to (3).

(3) Test for a "O" or a "D" (octal or decimal digits to be entered), skip to (43)

(43) Set A_6 false for octal, true for decimal. Skip to (40)

(40) Reset A_1 through A_5 and skip to (0)

Fill address of first register to be filled. As each character of the address is filled, A_1 through A_3, if octal or A_4 if decimal, is set up to correspond to the true binary code. A_5 is set false indicating a character. (A_5 is set true for a control symbol, such as a "D" or "O"). Set K false in (0) and count to (1)

(1) Test A_5. If false, skip to (41)

(41) Fill $A_1 \rightarrow A_3$ into E, shifting E 3 bits left; skip to (40)

(40) Reset $A_1 \rightarrow A_5$; skip to (0)

As each digit is entered, it shifts the proceeding one to the left in E.

(0) Enter dash, which indicates the previous number was an address, Count to (4)

(4) Test for dash; skip to (44)

(44) Read E into H; skip to (40) $\longrightarrow$ (0)

Select the number system the data is to be entered under. Once a number system is selected, it is preserved until changed. The system chosen is indicated by a light on the control console.

(0) Enter "D" if following number is not in octal; count to (3)

(3) Test for "D" or "O"; skip to (43)

(43) Set A_6 true for "D", and false for "O"; skip to (40) $\longrightarrow$ (0)

Proceed to fill in data

Type in instruction or number (operation or sign first) using blocks (1) ⟶ (41) ⟶ (40) ⟶ (0). After complete word is entered, press tab, which sets up $A_5A_4A_3A_2A_1$', and causes a skip from (5) ⟶ (45). Set up $L_5 \longrightarrow L_1$ to m_3 of H or address filled in Step 1. ($L_5 \longrightarrow L_1$ selects channel.) Count to (46) and search for m_3 sector. If L_5 is ~~true~~ false, then sector is on main drum and at coincidence between m_2 (permanently recorded sector coding) and H_2, turn R_1 true and count to (47), where E is recorded in M. (If L_5 was true, then E would record in J, the buffer register, using R_2.) Turn R_1 off. Count to (50).

(50) Add one to m_3 of H. Set K true; skip to (40) ⟶ (0).

After filling all commands and information, enter address of first command (in octal) in m_2 of E, transfer to H (dash) and press start (S). This counts (1) ⟶ (6) where $L_5 \longrightarrow L_1$ are set up as channel selectors for control number in m_2 of H; count to (7), find sector or word coincidence with M_w and m_2 of H. When found set K false and count to (10)

(10) Add one to m_2 of H (control #) and transfer H to G. Read into H instruction from M if L_5 is false or from J if L_5 is true. Test A_7 during O_{12} and O_{13} time for an overflow from the previous command. If an overflow had occurred and the command is not TO (test for overflow) or SL (shift logically) set K true and skip to (434) for printout and halt. If K is left false, count to (12). H now contains instructions and address of numbers

(12) Set $L_5 \longrightarrow L_1$ to channel of m_1; count to (13)

(13) Coincidence with M_w and H_2 m_1; count to (14)

(14) Read m_1 to E; count to (15)

(15) Set $L_5 \longrightarrow L_1$ to channel m2; count to (16)

(16) Coincidence with M_w and H_2 m_2; count to (17)

(17) Read m_2 to F and m_2 of G into H; ~~count to (11)~~

(11) Clear A_1 through A_7 and test H for a pulse in P_1 O_{13}. If there is a pulse (indicating a standard arithmetic operation) set K false and count to (20). If K is left true, skip to (400) for input-output commands.

17 - C# → H

~~ADD~~ (Command 35)

Add (m_1) to (m_2) and put results in (m_3)

(20) Set K true; add 1 to I (command) and count to (21)
[011 101 to 011 110 (35 + 1 is 36)]

(21) Set K true; add one (011 110 to 011 111). Count to (22)

(22) Set K true; remains true through P_2 O_{13} because command number has reached number 37. (indicates breakaway) Skip to (122)

(122) Set K false (E holds a command if O_{12} P_2 or O_{13} P_0, P_1 is a 1. This turns K on; skip to 126). If Number, count to (123); set A_1 true if E is -; set A_2 true if F is -

(123) Set K true if (E) or (F) is - (O_{12} P_1) and skip to (102); if not, count to (124)

(102) Complements negative number E or F as indicated by A_1 or A_2

(124) Add (E) to (F) sum in (E). Enter sign and overflow into E. Set A_7 true if overflow at O_{13} P2. If K is true, or (A_1 A_2 + A_1' A_2') is true before O_{13} P_2, skip out to (134). This indicates a true number. If not (A complement), count to (125)

(125) Complement E, set K true at O_{13}; skip to (134)

(134) Set $L_5 \longrightarrow L_1$ (channel selector) to m_3 of H, set A_1 and K false; count to (135)

(135) Find coincidence between M_w and $L_5 \longrightarrow L_1$. Turn R_1 true when sector is found. Count to (136)

(136) Read E to M. If A_{10} is false, set K true, set R_1 false; skip to (6). (A_{10} is true if two words are to be recorded)

If the number ^in E is a command, block (126) would complement F if necessary and count to (127)

(127) Add E to F sum in E but retain sign digits of E as sign of result and count to (134)

SUBTRACT (Command 36)

Subtract (m_2) from (m_1) put results in (m_3)

Count to (21) ending with K true; skip to (121)

(121) Reverse sign of F set K false; count to (122)

(122) Test for number or command and repeat operations as in ADD.

TEST MAGNITUDE (Command 34)

Compare magnitudes of (m_1) and (m_2). If (m_1) is greater than (m_2) take next command from (m_3); if not, take next command in order. Count to block (23); skip to (63)

(63) Compare E with F, and set A_1 true if (E) > (F) ^clear E? put marker pulse in P_0 O_{12} of E; skip to (65)

(65) If A_1 true, shift m_3 of H left into m_2 by following H_2 with $A_3 \longrightarrow A_4 \longrightarrow A_5 \rightarrow H_0$ for 4 word times. Shift E left 1 pulse each word time. When $P_D\ O_{12}$ marker reaches $P_1\ O_{13}$, turn K false; count to (66). If A_1 false, do not shift H_2 and set K false; count to (66)

(66) Set K true; skip to (6)

TEST ALGEBRAICALLY (Command 33)

Compare (m_1) and (m_2). If (m_2) is more positive than (m_3), take next command from (m 3), if not, take next command in order.

(23) Skip to (64)

(64) Test E and F for magnitude and sign, if E is more positive turn A_1 true and skip to (65)

(65) Same as above

TEST FOR OVERFLOW (Command 37)

Test (m_1). If it contains an overflow pulse take the next command from (m_3), if not, take next command in order.

(20) Skip to (60)

(60) Test $P_D\ O_{12}$ of E for overflow pulse. If there, turn A_1 true, put marker in E $P_D\ O_{12}$; skip to (65)

(65) Same as above

EXTRACT (Command 32)

Extract from (m_1) the binary digits which are in the same position as the "ones" in (m_2) and insert them in the same positions of (m_3). Do not otherwise change (m_3)

(25) Skip to (75)

(75) Set $L_5 \longrightarrow L_1$ up to channel locations of m_3 in H; count to (76)

(76) Find sector of m_3 and turn K false; count to (77)

(77) Read m_3 into G; count to (100)

(100) E_0 follows itself if F_2 is true, E_0 follows G_5 if F_2 is false, i. e., put bits of E into G when F has ones, with the result ending up in G. Trfr $G \rightarrow E$

(134) Locate channel of m_3 and proceed as in ADD routine

SCALE FACTOR (Command 31)

Shift (m_2) left until a binary one is in the MSBD of the magnitude. Subtract from (m_1) the number of shifts necessary. Put (m_1) into (m_3) and put (m_2) into cell following (m_3)

(26) Skip to (107) [E contains (m_1); F contains (m_2).]

(107) Transfer F into G, set K false and A_{10} true; count to (110)

(110) Test for pulse in P_2 O_{11} of G, if pulse, set K true and skip to (123). Set A_2 true, A_1 false; count to (111) if no pulse

(111) A_1 follow G_5, G follow A_1 and shifts left one, add one to F and make negative using A_2, set K true to skip out and reset to false at end of word; skip to (110)

(110) Repeat as above until a pulse in P_2 O_{11} of G or P_0 O_2 of F (signifying that F was all zeros), then skip out (123)

(123) Proceed as in ADD, (this subtracts number of shifts in F from (m_1) in E) except at block (136)

(136) Records E [(m_1) less number shifts in (m_3).] Transfer G→E. Count to (137)

(137) Record E into $(m_3 + 1)$. This records shifted data. Skip out to (6)

SHIFT MAGNITUDE (Command 30)

Shift (m_1) left or right as indicated by (m_2). The number of binary shifts is the magnitude of (m_2). If (m_2) is + shift left. If (m_2) is - shift right. Put results in (m_3)

(27) Skip to (132)

(132) Set A_2 true if F is -, set A_1 false. Subtract 1 from F, set K false first pulse in F. Look at A_3 to see if overflow occurred (Record in P_0 O_{12} E) Count to (133); skip to (134) if no ones in F_2

(133) If A_2 is false, shift E left following A_1 which follows E_5. If A_2 is true, shift E right following E_4 (Do not shift sign or command). If E overflows on a shift left, sign A_3 on. Skip to (132)

(132) Keep shift and reduce F till no pulse in F, then skip to (134)

(134) Same as in ADD routine

SHIFT LOGICALLY (Command 27)

Shift as above but shift entire length of (m_1)

(30) Skip to (130)

(130) Repeat as in (132)

(131) Repeat as in (133), except shift whole number including O_{12} O_{13}

MULTIPLY DOUBLE LENGTH (Command 26)
MULTIPLY ROUND-OFF (Command 25)

Multiply (m_1) by (m_2). Put least significant part of product in (m_3), the most significant in next cell after (m_3)

The multiplicand is in E and the multiplier in F. The product will be formed in G and at each successive addition, the LSBD will be shifted out of G and into E until at end of 36 shifts; the most significant part of product will be in G, and least in E.

(31) Skip to (171)

(171) Set K false and A_{10} true for double precision; count to (172)

(172) Put sign of E in A_4 of F in A_6, clear G, put 3 in O_{12} of H, test LBD of F (use E_4 to set up A_1) for first addition; count to (173)

(173) Shift G right, storing LBD in A_3. If A_1 true, add F to G, if false, do not. Use A_2 for carry flip-flop. Shift E right using E and put shifted out BD in A_1. Subtract one from O12 O13 of H. Test H for pulse, set K true; skip to (173) and repeat four times. At end of fourth time, no pulse in H, K false; count to (174) and put 011111 in O_{12} O_{13} of H.

(174) Repeat as in (173), except subtract 32 times from H $O_{12}O_{13}$ until all 0, then count to (175)

(175) Shift E and G left, put shifted off digit of G into A_3, then into E at P_2 O_{11}. Put sign of product in E and G. Count to (176) if A_{10} true, then skip to (134) if A_{10} false count to (177)

(177) Round off G using A_3 and put results in E. Set K true; skip to (134)

DIVIDE DOUBLE PRECISION (Command 24)
DIVIDE WITH ROUND OFF (Command 23)

Divide (m_1) by (m_2) and record the quotient rounded off in (m_3). In a normal divide operation the dividend is in the E register and its magnitude is smaller than that of the divisor which is held in the F register. The quotient then is less than unity and is generated in the G register. If E is greater than F an overflow occurs which, unless programmed for will cause the computer to halt computation.

Divide is accomplished by testing the magnitudes of E and F. If E is smaller than F, E is shifted left (binary division begins at the MSBD) and a 0 is recorded in the quotient register. G is then shifted left for the next generated quotient pulse. F is not shifted. If E is larger than F, F is subtracted from E and the result is recorded in E. A one is set in G. Both G and E are shifted left. In either case the shifted E is compared with F to determine whether the next operation should be a subtraction or not.

In the 102-A five operations are carried on simultaneously during one word time in divide. These are: Subtract F from E, shift E, record pulse or absence of pulse in G, shift G, compare F with shifted E.

These operations are done in the following logical way. On command to divide, count to block (34); skip to (114)

(113) For double precision set A_{10} true.

(114) Compare E and F. Set A_1 true if F greater than E, false if E larger than F. Store sign of E in A_4, sign of F in A_6 until needed at end of divide. Set A_7 false if E is equal to or greater than F. (This indicates an overflow and will be used later.) Put a marker pulse in P_0 O_1 of G to indicate the end of division. Set K false and count to (115)

(115) If A_1 is true $(F > E)$ do not subtract F from E and enter 0 into G. Shift E and G left, compare F with shifted E using A_8. E is shifted left through A_5, G is shifted left through A_3. If A_1 is false $(E < F)$ subtract F from E and shift results left comparing remainder with F. Record a 1 in G and shift left. At end of the word time test for marker in G at P_0 O_{13} to indicate end of division. If no pulse, set K true and skip to beginning of block. In addition at end of block set control flip-flop A_1 to configuration of A_8 which indicates whether $E >$ or $< F$. Repeat block (115) subtracting F from E if A_1 is false and shifting the results left, putting a 1 in G; shifting E left without subtracting if A_1 is true and putting an 0 in G until the marker pulse originally set in P_0 O_1 of G is sensed at P_0 O_{13}. Then set K false and count to (116)

(116) Test for double precision (A_{10} on). Set A_3 true if MSBD in E is a 1 for round-off purposes. Put the sign of the quotient into G and sign of dividend (A_4) into E. If A_7 was on record an overflow pulse in G and E. If A_{10} was true skip to (134); if false, count to (117)

(117) Round off the quotient by adding A_3 to G and putting results in E. Set K true and skip to (134)

(134) Proceed as in the ADD routine for put-away.

HALT ~~Command~~ 22

Count to block (35)

(35) Set K true and skip to (0)

NO COMMAND

If there is no command given, count to block (37)

(37) Set K true and skip to (434)

(434) Read G into E and clear G. (E now contains control number) Put P_2 O_{12} pulse in F. (This marker signifies that after first print-out return to (0)). Recirculate m_1 and m_2 of H. (Eliminating m_3 of H prevents all but one word print out.) Skip to (214)

(214) Proceed as in print to block (233)

(233) ~~Test for error. (Marker pulse in P_2 O_{12} of F)~~ Set K true and skip to (0)

PRINT

Type out (m_1) and next N cell as indicated in (m_3) in the mode of printing as indicated in (m_2)

The address cells of the print command contain:

m_1 = address of word to be printed
m_2 = sign position contains mode of printing and the magnitude. (The number of digits to be printed.)*
m_3 = number of words to be printed after the one in m_1

(35) Skip to (236)

(236) Test to see whether octal address are to be printed, indicated by a $P_1 O_{12}$ in F which contains m_2 of command. If no pulse, skip to (210) for number printout. If pulse, set K false and count to (237) for address and number printout.

(237) Put marker pulse $P_0 O_{11}$ in G to indicate digit to be printed and read H into E. Count to (240)

(240) Set K and A_4 false. Set octal digit to be printed in A_1 - A_3. A_1 follows (G_5)(E_3); A_2 follows (A_7)(E_5); A_7 following G_5; and A_3 follows (A_8) (E_5); A_8 following A_7. In this manner, when the marker pulse in G is sensed at the beginning of the digit to be printed, the first binary bit is set in A_1, and A_7 is set true. With A_7 true, the next bit is put in A_2, and A_8 is set true. A_8 true puts next bit from E_5 in A_1. Count to (241).

* See 102-A descriptive literature for various modes of print-out

(241) Set up H for delay in next block by filling 1's in P_0O_{12} P_1O_{13}. Set A_1 - A_6 in Flexowriter code from logic developed from previous set up of A_1 - A_4; count to (242)

(242) Actuate the Flexowriter key corresponding to the code set up in A_1 - A_6. Subtract one from O_{12} O_{13} of H. If there is a pulse in O_{12} O_{13} of H, set K true and repeat block setting K false at beginning of block. This gives a 32 word delay [to insure Flexowriter solenoid is actuated] before K is left false. Then count to (243)

(243) Set A_5 and A_6 false, K true and repeat block until Flexowriter signal sets K false, signifying end of print; count to (244)

(244) Test for last digit of address. The marker will be in P_0O_8 if it is. If it isn't, count to (245)

(245) Set K true and shift G right one octal digit and skip to (240) to repeat printout

(244) If K set true, skip to (204)

(204) Set A_1 - A_6 to Flexowriter code for space. Set K false, count to (205)

(205) Fill in H, O_{12} O_{13} for 32 word delay in next block, count to (206)

(206) Actuate Flexowriter to space as coded in A_1 - A_6 and repeat block until O_{12} O_{13} of A is zero, as in block (242) and then count to (207)

(207) Reset A_5 and A_6 false and wait in block for Flexowriter ready signal. Then count to (210) (Address has now been printed)

(210) Set up L_1 - L_5 to find m_1, set K false and count to (211)

(211) Find sector of m_1, and when found, count to block (212)

(212) Read M into E, test P_0O_{13} of F_2 for pulse which indicates alphabetical print or not. (m_2 of command in F.) If not, leave K false and count to (213)

(213) Test P_0O_{13} of F_2 for pulse indicating decimal or octal. If no pulse, set K false for octal and count to (214) skip to 253 for decimal

(214) Put marker pulse in G, P_0O_{13}

(215) Set up A_1 - A_3 for octal digits, A_4 false, K false and proceed through blocks (216), (217), (220) as in blocks(240), (241), (242) and (243) previously. Count to (221)

(221) Test in F for last digit to be printed (F set up by code pulses in m_2 of command), if not, count to (222), shift G right, and skip to (215). If last digit printed, set K true and skip to (225)

(225) Set K false, set A_1- A_6 to Flexowriter tab code and count to (226)

(226) Proceed through blocks (226), (227), (230), as in blocks (206), (207) and (210); count to (231)

(231) Add 1 to m_1 of H, making it the new address of the next word to be printed out, subtract one from m_3 of H. (m_3 being the number of words to print out) K is used for adding and subtracting. Count to (232)

(232) Test m_3 for zero. If last word has not been printed ($m_3 = 0$) set K true and skip to (236). If yes, count to (233).

(233) Test P_2O_{12} of F for an error. (There will be a pulse if coming from (434); any error causes a skip to (434)) If an error, set K true and skip to (0) if not, count to (234)

(234) Set K true and skip to (6) for next operation

Note: Octal-digits print routine prints out sign in octal notation

If the command had been to print a decimal digit (213) would skip to (253). A subroutine must be used previous to printout command to convert octal to binary-coded decimal.

(253) Put marker P_0O_{12} of G for decimal printing. Set K false and count to (254)

(254) Set sign of decimal digit into A_1 - A_4 using A_7 - A_9 as in block (240); count to (255)

(255) Set up A_1 - A_6 in Flexowriter code for +, -, P, N, and set up delay pulses in H. Count to (256)

(256) Actuate Flexowriter and repeat for 32 word times, count to (257)

(257) Wait for Flexowriter, proceed signal and set A_5, A_6, false; count to (260)

(260) Shift G four pulses right, count to (261)

(261) Set up A_1 - A_4 as in (254); count to (262)

(262) Set up A_1 - A_6 in Flexowriter code for decimal digit and proceed as in blocks (215) (221), testing in block (265) for last digit and if yes, skipping to (225) and finishing print routine.

If the command was to print alphabetical, it will be necessary to set up A_1 - A_6 to correspond to two octal digits, since it takes two digits to represent one alphabetical character. Also it will not be necessary to convert to the Flexowriter code since each pair of octal digits will already have been programmed to represent the proper code. First skip from (212) to (250)

(250) Put marker pulse in P_0 O_{11} of G. Skip to (270)

(270) Set up $A_1 - A_3$ from O_{11} of E using $A_7 - A_8$ as in previous blocks. Set K false and count to (271)

(271) Shift marker in G three pulses right. Count to (272)

(272) At P_0 O_0, set $A_4 - A_6$ to configuration of $A_1 - A_3$ and set up $A_1 - A_3$ to new digit (O_{10}). Set up H for delay in next block; count to (273)

(273) Proceed as in blocks (242) (276). If the last digit has been printed, skip from (275) to (231), if not, skip from (276) to (270)