PDP-1

INPUT-OUTPUT SYSTEMS MANUAL

(PRELIMINARY MANUAL)

F-25

INDEX

INDEX

THE ELECTRICAL INTER-CONNECTION AND PROGRAMMING FOR DEVICES CONNECTED WITH PDP-1

Introduction

This is a discussion of the electrical, physical, and programming aspects of devices connected to PDP-1.

The PDP-1 is composed of standard DEC building blocks whose logical characteristics and capabilities are discussed in DEC literature.* The following comments on the inter-connection of equipment will assume the reader has a fairly basic knowledge of Boolean Algebra and has vaguely perused the DEC Logic Handbook, A-400-B. The reader should also be familiar with the register layout of the PDP-1 and its programming as described in the Programmed Data Processor manual, F-15A.

Information and Control

Two general types of signal flow are in PDP-1. These are for information transfers and for control pulses. An example of an information transfer in PDP-1 is when the contents of the Memory Buffer register (MB) are read into the Accumulator (AC). In this case, when the transfer is made, a whole register, or 18 bits of information, is transfered simultaneously. The command that the Memory Buffer register is to be placed in the Accumulator is done by a single line which is called a control line. That is, the 18 bits of the Memory Buffer register go to the Accumulator, and the control line "samples" these lines at an appropriate time.

Basically, external information controls are handled in the same manner. Information is presented to external devices, and the device is given a control signal under program control which says to take the information and proceed. A symbolic language program may be written for the paper tape punch to perforate one line of paper tape:

```
character     77                  ,code to be punched on tape
               .
               .
               .
punch         lio character       ,contents of "character" replace
                                   In Out register
              ppa                 ,command to punch one line of tape
```

In the above program, if the program is started in register, "punch", the instruction, lio character, would place the octal

77, in the In Out register (IO). The ppa or punch paper alphanumeric instruction would transfer the last 8 bits of a character, 00 111 111 (77), to the punch logic. The punch would then perforate a line of the tape. Information is transfered to paper tape just as information is transfered from a memory location to the arithmetic element. The character 00 111 111 would correspond to the information and any control signals generated by the command punch paper alpha (ppa), being given would constitute the control to the punch logic.

The in-out transfer command has the operation code 72XXXX. The address portion of the command has special meaning. The six bits, 12-17, address one of 64 devices. Bits 5 and 6 control synchronization and bits 7-11 may be used for special purposes.

The in-out transfer command is the basic method of transfering information between PDP-1 and other devices.

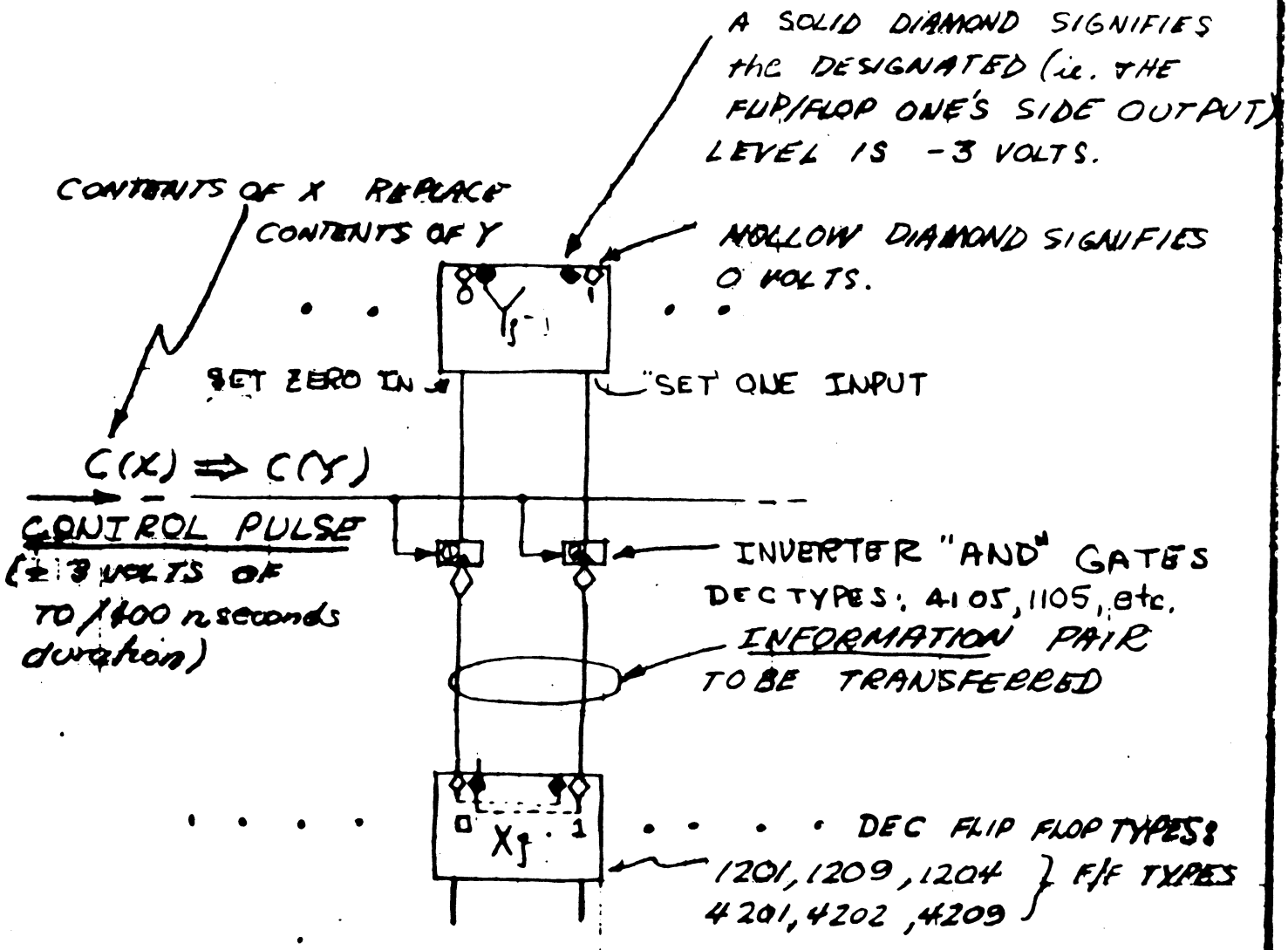
There are several methods (some of which are special options) of transfering information that relieve the program of the details associated with the transfer. Logic within a device may request that information be transfered directly from core memory while the program is running. That is, the program is momentarily interrupted while data are transfered. This is called a high speed data channel.

The Sequence Break System permits a program sequence to be interrupted at a time a condition has been satisfied. The Sequence Break relieves the program of the details associated with the status of input-output devices.

Fundamental Transfer of Information

Fundamentally, information can be transfered between one register and another register, as shown in figure 1. This method, which involves only one step is known as a jam transfer. Figure 1 shows the jth flip-flops for registers Y and X. At a given time, information is to be transfered from register X to register Y. The control event is entitled "C(X) = C(Y)", or, the contents of register X replace the contents of register Y. A pulse on the control line will transfer the information from register X to register Y by "sampling" the output voltages of register X. This process may also be referred to as "strobing" , "sampling", "transfering" or "reading in".

If register X contains a one, then the transistor "and" gate labeled 2 will conduct when the control pulse is given causing the



JAM TRANSFER OF INFORMATION BETWEEN TWO REGISTERS

FIGURE 1

set one input to be pulsed. Transistors 1 and 2 act as "and" gates. The new contents of Y do not depend on the previous contents. The contents of X are unaffected by the information transfer. The logical conventions of figure 1 will be used throughout DEC PDP-1 diagrams.

Figure 2 shows the two step method of transmitting information between two flip-flop registers. By allowing the transfer to be effected in two steps, the number of "and" gates has been halved.

First, each bit of the register is set to zero. A short time later (0.2 microseconds for 5 MC logic and 2 microseconds for 500 KC logic), a second pulse transfers (strobes, or reads in) the information into the one side of the flip-flops. Using this system, information need only be specified by the polarity on a single line rather than two lines.

The Use of In-Out Transfer Commands For Control Purposes

The PDP-1 uses the address of in out transfer (iot) instructions to select various devices. Actually, the decoding for the instruction, 72XXXX, is as follows:

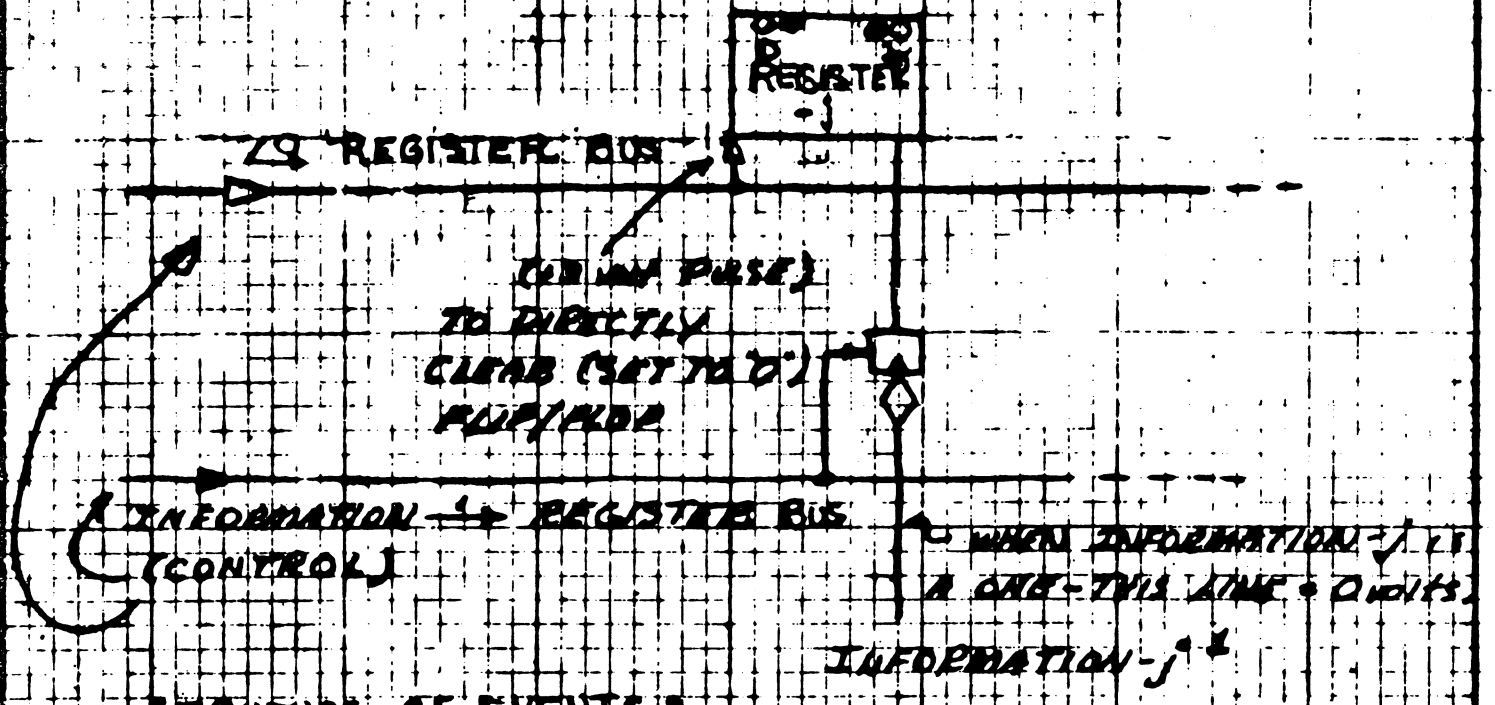
Bits	Use
0-4	11101 Instruction bit code for in out transfer
5-6	Used for device synchronization
7-11	Unused - May be anything
12-17	Addresses 1 of 64 devices

The address decoding scheme will be described in greater detail in the following sections. One of 64 pulses will be emitted on a particular wire corresponding to the address or last 6 bits of the in out transfer instruction.

Really, 64 pulse pairs may be formed. The pulse pair, which corresponds to an address is sent on two separate wires and provides the following:

1. 2.5 microseconds after the command is given, a pulse is available for clearing a register.

2. 2.5 microseconds later, or 5 microseconds after the beginning of the command, a pulse is available to transfer the information. Thus, the two step transfer method may be used.



SEQUENCE OF EVENTS:

1. REGISTER IS CLEARED (LD REGISTER)
2. INFORMATION IS TRANSFERRED (STROBED/SAMPLED) INTO REGISTER (INFO → REGISTER)

TWO STEP METHOD TO TRANSFER INFORMATION TO A REGISTER.

FIGURE 2.

For example, an in-out transfer command, turn on (ton), may be defined in the program assembly language and connected within PDP-1 with the operation code 72 0010. Now, each time a program gives a ton command, a pulse of 0.4 microseconds duration and 3 volt amplitude will be emitted on a line labeled "ton". Similarly, a turn off pulse may be emitted for a tof command (code 72 0011). The following program with figure 3 forms a 50 KC square wave generator:

```
on      72 0010          ,sets the flip-flop to a one state
        jmp off          ,dummy instruction - goes to next
                         ,instruction
off     tof              ,sets the flip-flop to the zero state
        jmp on           ,returns to the register labeled begin
                         ,for repeating the sequence
```

A pulse appears on the ton line at some time. Five microseconds elapse as the "jmp off" command is effected. Five microseconds later, tof occurs and a pulse clears the flip-flop. The jmp instruction requires 5 microseconds and the process repeats. Every 10 microseconds, the flip-flop either gets set or cleared. Thus, output of the flip-flop is a square wave with a period of 20 microseconds. The times for an on or an off level can be varied in 5 microseconds intervals.

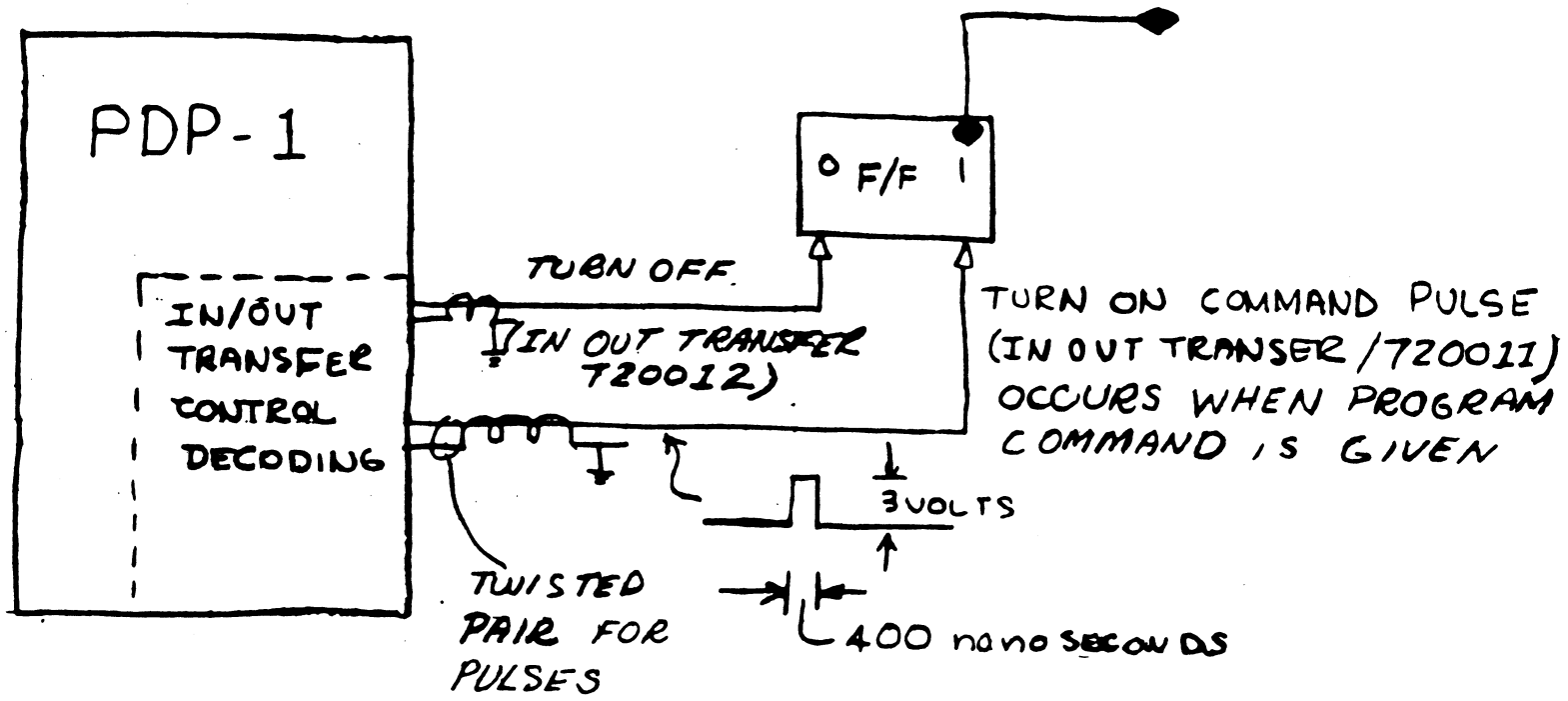
TRANSMITTING INFORMATION TO A DEVICE WITH THE PROGRAM

One of the most common devices that can be connected to the PDP-1 is a flip-flop register of up to 18 bits. The register, for example, might drive a digital to analog decoding network to supply an analog voltage, or a series of relays, or a visual display buffer from which decimal numbers are decoded and viewed, etc.

Given, there is 18 bits of information in the in-out register, we wish to transfer these bits (information) to an external register.

Figure 4 shows information of the In-Out register Bit-j being transmitted to the 18-bit external buffer register (EB). To transmit information to External Buffer Register, an in out command address is assigned which we will call teb, with operation code 72 0010. The one's side outputs of the In-Out register (IO), are sent to EB. When a bit of the IO is a one then the respective line is 0 volts, and when the IO register bit is a zero, the line is at -3 volts. When the program gives the teb, two things happen:

1. 2.5 microseconds after the command begins, a positive pulse clears EB.



PDP-1 WITH TWO CONTROL PULSES

FIGURE 3

2. 2.5 microseconds later, or 5 microseconds after the beginning of teb, the information from the one's side outputs of the in-out register is read into (or transfered to) EB. Thus, 5 micro-seconds after the beginning of teb, EB contains the same information as IO.

A block diagram of the connection is shown in figure 4B. There are 18 lines of In-Out register information, and 2 control pulses. Each of the control pulses requires either a coaxial cable or a pair of wires which are twisted, thus, a total of 22 wires form the inter-connection.

RECEIVING INFORMATION FROM A DEVICE WITH THE PROGRAM

When information is received from a remote device, the roles of transmitter and receiver mentioned above are reversed. A register of information must be transmitted to PDP-1, as in the case of an analog to digital converter attached to the PDP-1.

Figure 5 shows how one bit of information would be read into the in-out register under program command. In this case, a switch position is to be sampled, and 18 switches form a register. In this case, a program executes a command which emits pulses that:

1. Clear the in-out register.

2. Sample the information and place it into the in-out register.

The mechanism for the information gating is shown in figure 5. That is, a level enables a capacitor diode gate, then a control pulse "ands" with the level to conditionally form a pulse for each information bit, thus, the IO register flip-flop is set to a one if the gate is enabled.

The Connection of An Analog To Digital Converter With PDP-1

The connection of a 12 bit analog to digital converter to operate with PDP-1 is shown in figure 6. The line labeled "start convert" is a control pulse to the converter that commands a conversion. From the time the "convert" command is given, the converter requires 40 microseconds to determine a 12-bit digital number proportional to the analog input voltage. The convert command, cnv, has the operation code 72 0041.

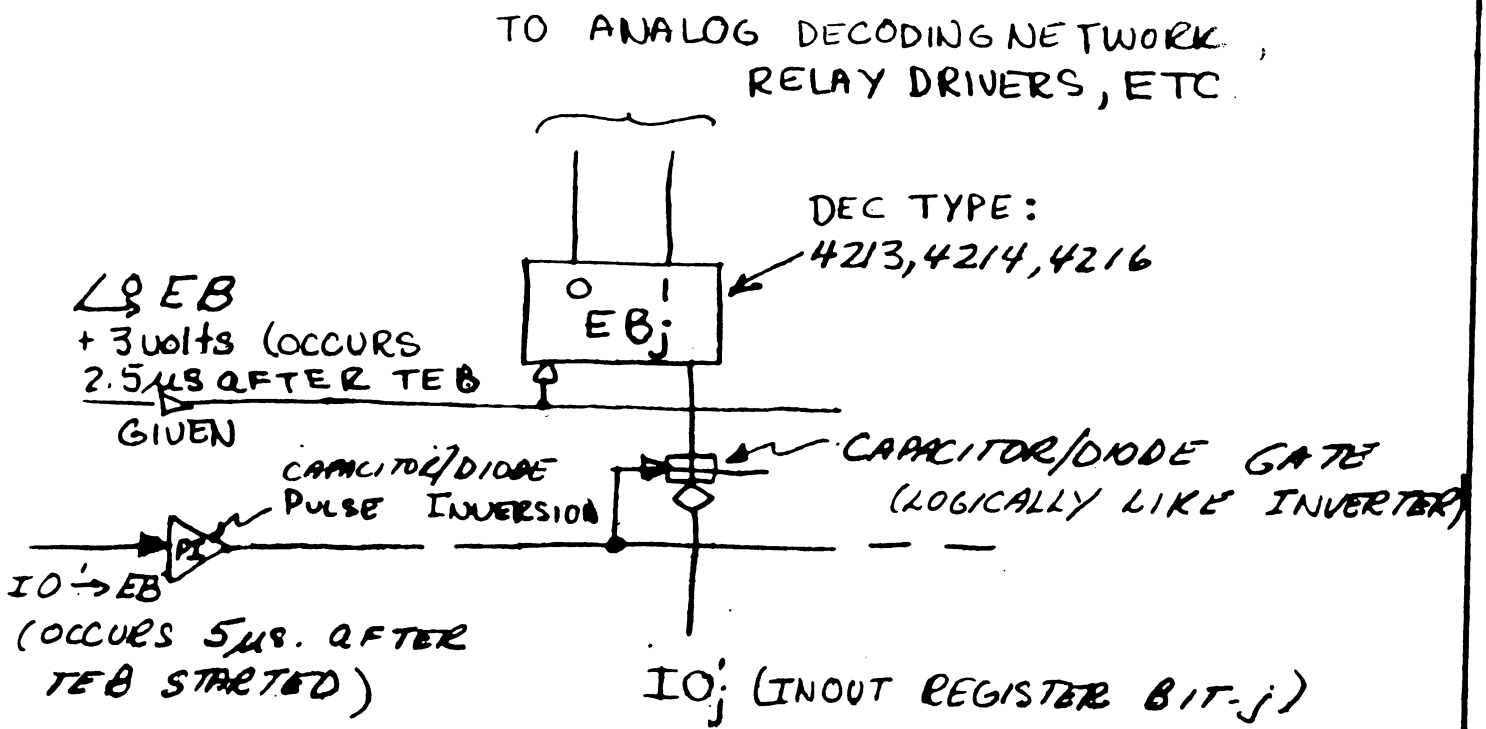


FIGURE 4a

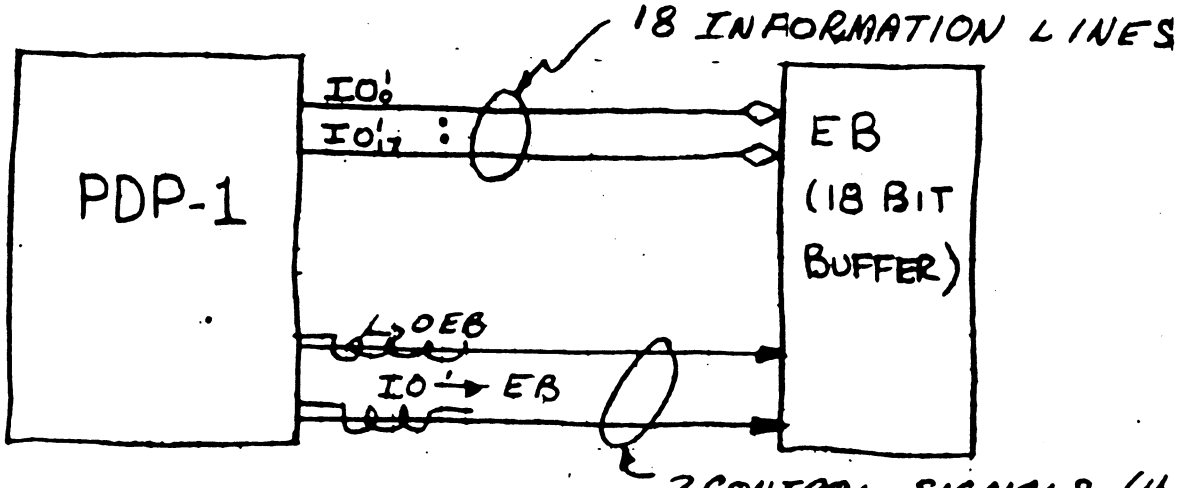
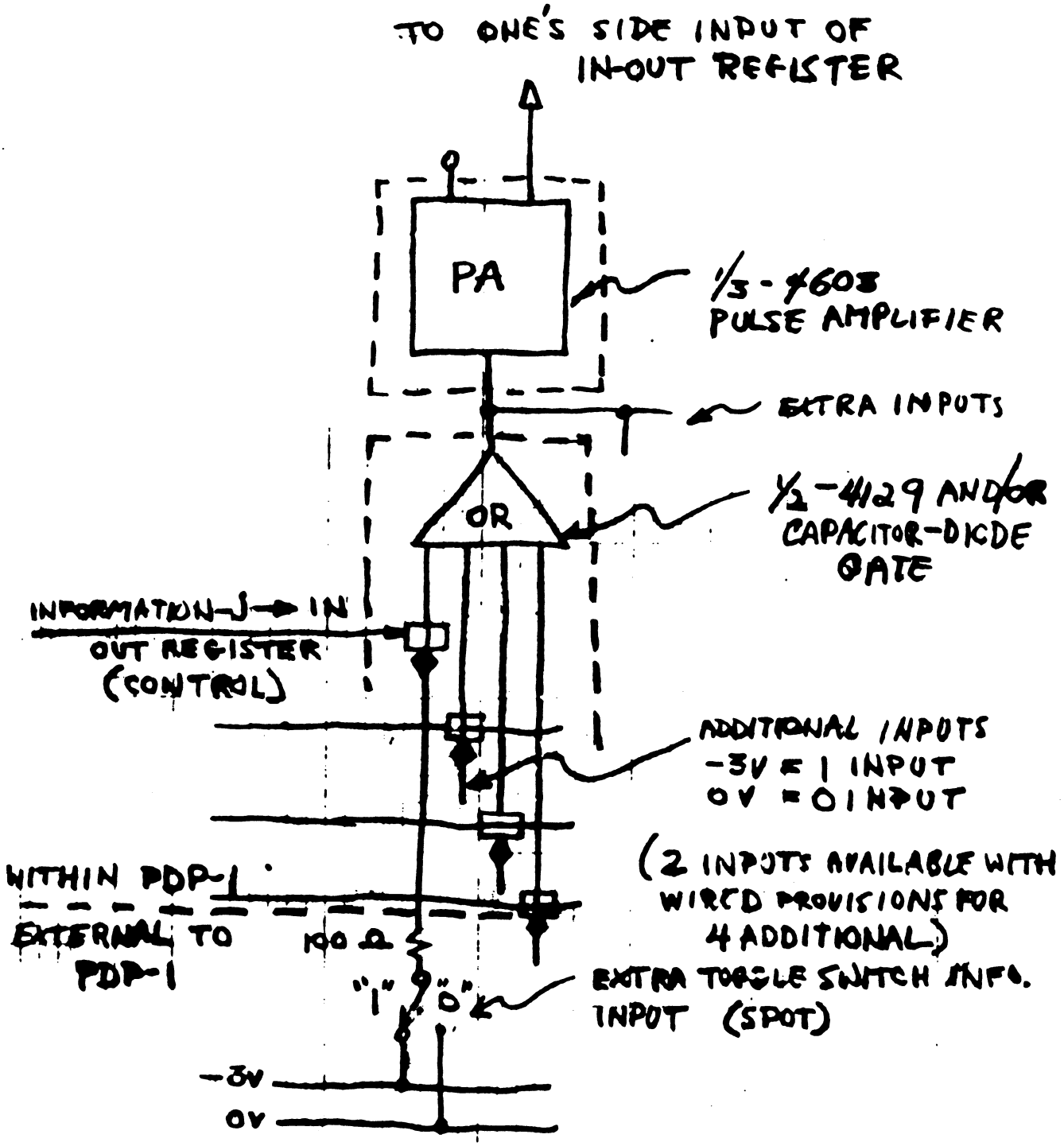


FIGURE 4b

TRANSMITTING INFORMATION TO EXTERNAL BUFFER (EB) USING COMMAND: TRANSFER TO EB (TEB)



IN-OUT REGISTER INPUT MIXER FOR ONE BIT

FIGURE 5

The digital number is transfered from the converter to the In-Out register under program control. The command, read converter buffer, (rcb = 72 0031) reads the converted bits into the 12 most significant bits of the In-Out register.

Figure 6 shows the decoding within PDP-1 necessary for control. The information is read into the input mixer (which sets the In-Out) and three pulse lines are decoded which:

1. Clear the in-out register (clear portion of rcb).

2. Sample the converter buffer outputs to set the various in-out bits (sample portion of rcb).

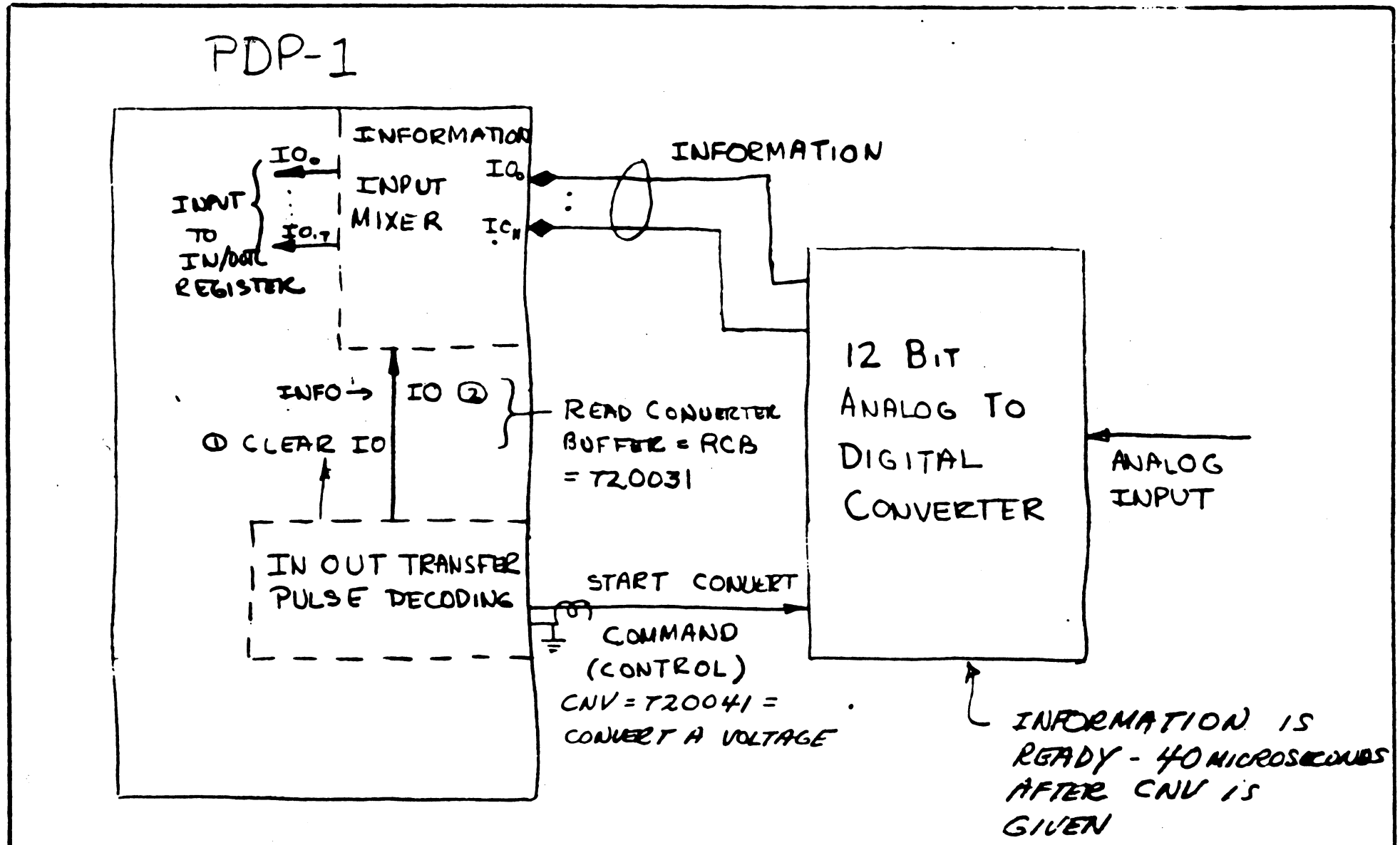
3. Start the conversion (CNV).

Pulse (2) occurs 2.5 microseconds after pulse (1) above.

The following subroutine uses the analog to digital converter, and is called by the command "jsp fill". The subroutine in symbolic assembly language samples an analog input 512 times at a 20 KC rate, and stores the results in registers: "function", "function & 1", ... , "function & 511".

```
,Subroutine to sample a continuous voltage 512 times, each 50
,microseconds and store in "function" table.  The subroutine starts
,at register "fill".

fill          dap exit
              law function - 1
              dap storsam          ,initialize
loop          cnv                  ,starts converter = 72 0041
              nop
              idx storsam
              sad ftest
exit          jmp                  ,exit location
              rcb                  ,rcb = 72 0031 place voltage sample
                                   ,in IO
storsam       dio                  ,store sample
              jmp loop
ftest         dio function + 1000
function      0                    ,first sample
   .          .
   .          .
   .          .
function +777 0                    ,1000₈ sample
```



PDP-1 / A-D CONVERTER CONNECTIONS

FIGURE 6

OPERATION OF THE STANDARD IN OUT EQUIPMENT

Synchronization and Programming

The straight forward use of the in-out transfer commands, e.g., the connection of analog to digital or digital to analog converters has been described above. In these cases, the transfer of information took on a very simple form since the program controlled the information transfers relative to the action of the device.

Quite often, input-output operations must be synchronized. That is, information is transfered certainly and efficiently, which may cause the computer (or a device) to "wait", and then proceed in synchronism. In other cases, when several in-out devices operate simultaneously, the synchronization is essential. The synchronization of the standard in-out equipment (oscilliscope, paper tape punch, photoelectric tape reader, and typewriter) is done simply and the control for this synchronization is coded in each in-out transfer command. Appendix II contains a current list of the assigned in-out transfer commands for the PDP-1 equipment.

Bacally, the program must always operate faster than a device. Thus, a program can halt, then continue in synchronism with a device. For example, the command to punch a line of paper tape is given, the paper tape punch effects the punching when able, then signals the waiting computer to proceed. Sometimes it would be desirable for the computer to give the command to punch a line of paper tape and not stop but continue calculations. This method is certain unless a second command is given before the punch has finished the previous task.

A safe method would issue a command, proceed with a safe number of calculations, then issue a waiting command which re-synchronizes the completion of the in-out transfer and the program. The re-synchronization takes place in 5 microseconds intervals.

The decoding for the in-out transfer command which allows various possibilities is shown in Table I:

TABLE I

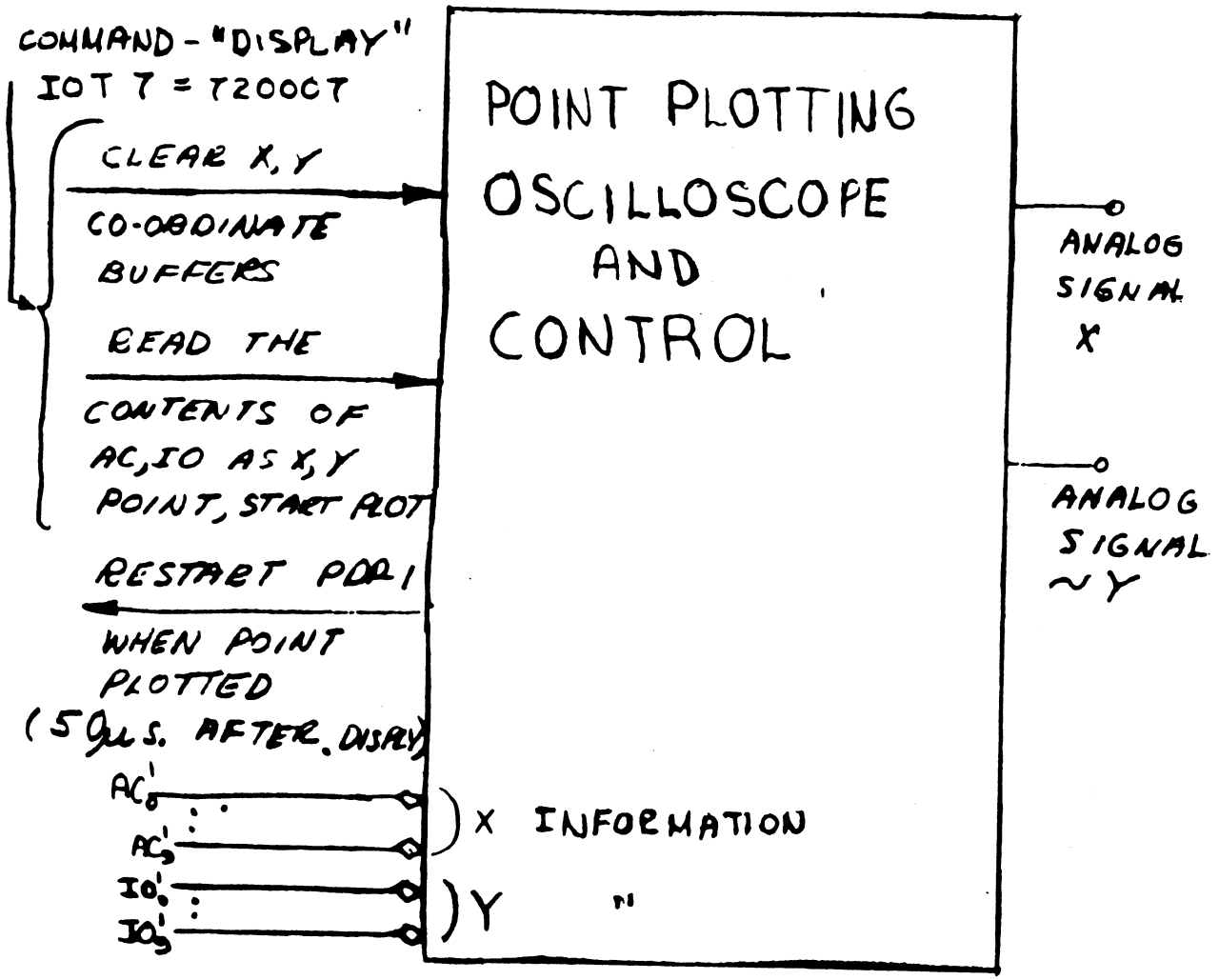
In-Out Transfer Command Bits 5	6	Wait for Completion Pulse for Restart/Continue without wait	Enable/Disable-Completion (Done Pulse Signal
0	0	Continue, no wait	Disable
0	1	Continue, no wait	Enable
1	0	Wait, then continue	Enable
1	1	Wait, then continue	Disable

Bit 5 of the in-out transfer command disignates whether the program is to wait for a completion pulse before continuing. The exclusive or of bits 5, 6 of the command specify whether the completion pulse return signal is to be enabled or disabled.

Point Plotting Oscilliscope

Figure 7 shows the diagram of the scope inter-connections to PDP-1. The oscilliscope operates on a point by point basis and is activated by the computer giving the command, "display", iot 7, or 72 0007 (or 73 0007, etc.). The display command first clears a 10 bit X coordinate buffer, a 10 bit Y coordinate buffer and 2.5 microseconds later reads the contents of the 10 most significant bits of the Accumulator and In-Out registers into the X and Y buffers and then intensifies the point. The plotting of a single point requires 50 microseconds. The cathode ray tube has a P7 phosphor, thus the point persists for a relatively long time. At the completion of the plotting, a pulse will be emitted on the restart line to PDP-1 which is used for the computer restart. The restart pulse occurs 50 microseconds after the display command is given.

The following program shows how the various in-out transfer commands are used to effect savings in timing, and handle synchronization. The first program displays a horizontal line at some Y coordinate, 1/2 the width of the scope. The line consists of 512 points, and is plotted starting at the point X = 0 going right to the point X = 377_8. The program requires (50 + 20) x 512 microseconds. This program uses the most straight forward in-out transfers. The "display" command, 73 0007, is given and the computer waits until the point is displayed before the machine restarts and continues calculations.



OSCILLOSCOPE FOR PDP-1

FIGURE 7

```
,Display a horizontal line of 512 points
,line plotted center to right edge - height of Y
start       lio y              ,y holds height at line
            cla
loop        730007             ,display a point and wait till done
            add increment
            sma
            jmp loop           ,return until ac = 400000
            --                 ,done
            .
            .
increment   400                ,increment of line
y           --                 ,y coordinate
```

The following program computes while the point is plotted:

```
start       lio Y
            cla
            jmp loop & 1
loop        730000             ,dummy - halts till previous completion
            724007             ,displays a point/computer proceeds
            add increment      ,display completion enabled
            sma
            jmp loop
            .
            .
            .
```

The above program plots the 512 point line in (50 + 5) x 512 microseconds. The program commands a point to be displayed, (without disabling the completion pulse), proceeds with the calculations, then finally gives a synchronizing "wait" instruction which synchronizes the display.

The only case not included above, but mentioned in Table I, is the case of bits 5 and 6 both zero. Here, the program does not halt and the completion pulse of the device is disabled. Thus, the program must only refrain from giving the display commands too frequently.

The technique of inhibiting completion is used when several devices are operating concurrently. The inhibit command completion is used with the sequence break system.

Paper Tape Punch For PDP-1

A block diagram of the paper tape punch logic is shown in figure 8. The punch action is similar to display. Two separate commands are given which transfer information to a buffer register for the paper tape and they are:

1. Punch paper alphanumeric format, (ppa) 72 0005, (or 73 0005).

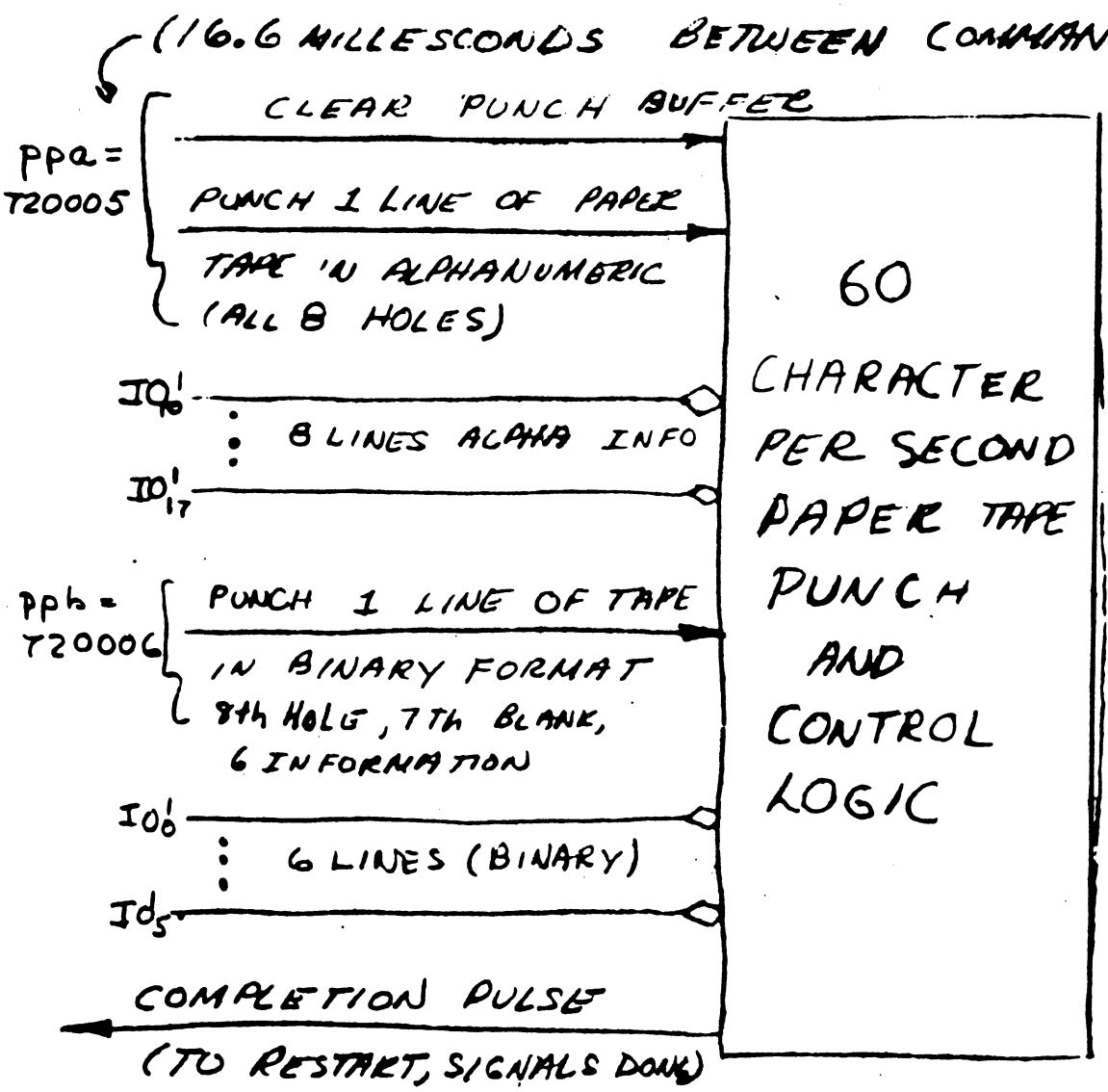
2. Punch paper binary format, (ppb) 72 0006, (or 73 0006).

When either the ppa or ppb command is given, a pulse first clears the punch flip-flop buffer register then 2.5 microseconds later a pulse will occur on either the punch paper alpha line or the punch paper binary control line and the punching action is initiated. Alphanumeric information consists of in-out register bits 10 to 17 for the 8 holes on paper tape. The ppb command always punched hole 8, ignore information for hole 7, and punched IO bits 0 - 5. A pulse occurs when the punch has finished an assigned task. This may occur within 4.0 milliseconds after the command is given.

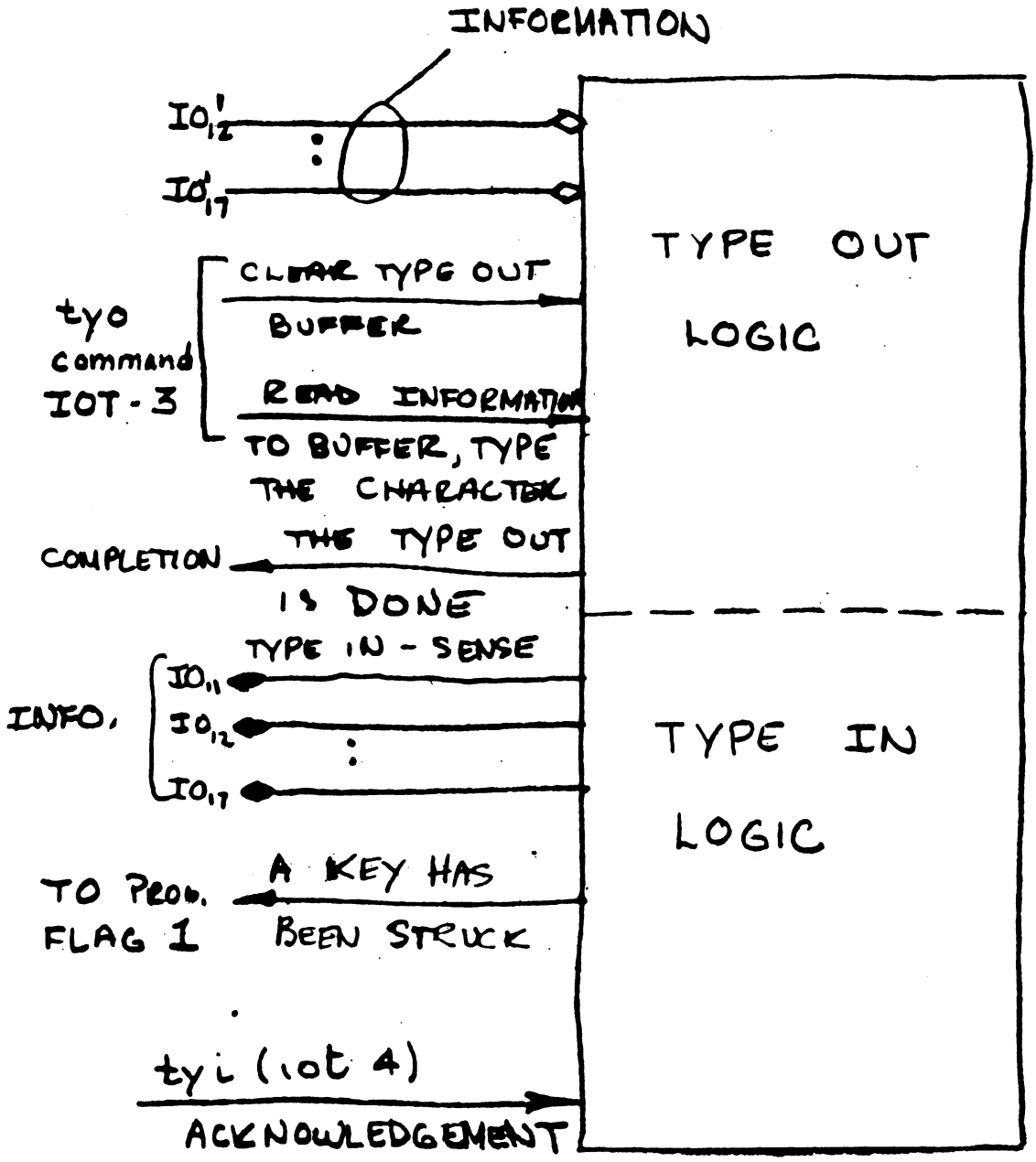
The paper tape punch is synchronous, and only during certain intervals can a character be punched. This condition is sensed by the control logic when the punch cams reach a "start" position. If a character is to be punched, information must be static for 4.0 milliseconds after the "start" punch position and solenoid driving current are enabled. The completion pulse is returned when the punch buffer is free to receive the next character. Thus, if a punch paper command is given and it appears at the correct time, the punching appears to require only 4.0 milliseconds. If a punch command is given and the punch has just passed "start", (the synchronizing position) the completion pulse may require 4.0 + 15.8 milliseconds.

Typewriter for Output of Information

The input-output typewriter for PDP-1 is shown in figure 9. The typewriter is best explained by separating the logic into output and input. For typing out, the typewriter acts like the display or the punch. The type out command, tyo or 720003, (or 730003) first clears the typewriter buffer then 2.5 microseconds later, the information of IO bits 12-17 is read into the typewriter buffer, and the typing action initiated.



PAPER TAPE PUNCH FOR PDP-1
FIGURE 8



TYPEWRITER FOR PDP-1

FIGURE 9

The procedure for typing a character would be:

1. Load the in-out register bits 12-17 with the code for the character to be typed.

2. Issue the command tyo.

The typeout portion of the logic has a completion pulse which is emitted when a character has been typed. The typewriter output rate is 9.5 characters per second, thus, the typeout completion pulse occurs approximately 105 milliseconds after the tyo is given.

The typeout completed signal can be used to restart the program if the typewriter is used synchronously within a program. The completion pulse, of course, can be used in the same manner discussed with the display system. The time saving is emphasized more in the case of the typewriter because the 105 ms interval between characters will allow up to 20,000 operations.

Typewriter Input

The typewriter input logic is also shown in figure 9. This logic is similar to the sampling of an analog to digital converter. When a key is struck, a pulse is emitted on the line labelled "A key has been struck". This pulse is wired to set program flag 1. A 6 bit buffer (the same used for type out) holds the code for the character struck. When flag 1 is set, the program may give the command, type in, tyi = iot 4 (72 0004) never 73 0004. The command, tyi, first clears the IO then reads the 6 information bits, which code the character typed, and the type in sense level into IO bits 11-17. Tyi first clears the in-out register, then 2.5 microseconds later the 7 levels of the typewriter are read into the IO.

The tyi command notifies the typewriter logic that the character has been accepted by the program. The type in acknowledgement resets the type in sense line, thus, if another character is not typed and the program gives the type in instruction, bit 11 will be a 1 instead of 0. In this way, a program can tell if the same character has been read more than once.

The following program excerpt uses tyi. The program begins in register "look".

```
look        szf * 1          ,650001 skip on flag set
            jmp look         ,repeat look, look + 1 till key struck
            tyi              ,720004 IO contains code for key struc
```

Instructions "look" and "look + 1" are repeated over and over again until the program flag 1 is set by a key being struck. When the key is struck, the program flag 1 is sensed and the instruction in look + 2 is obeyed, tyi, and the 6 bit code for the character typed is placed in the in-out register. If the key typed was a carriage return, then an octal 77 would appear in the IO. If a second tyi command is given before a new character is struck, the in-out register would contain the code 177.

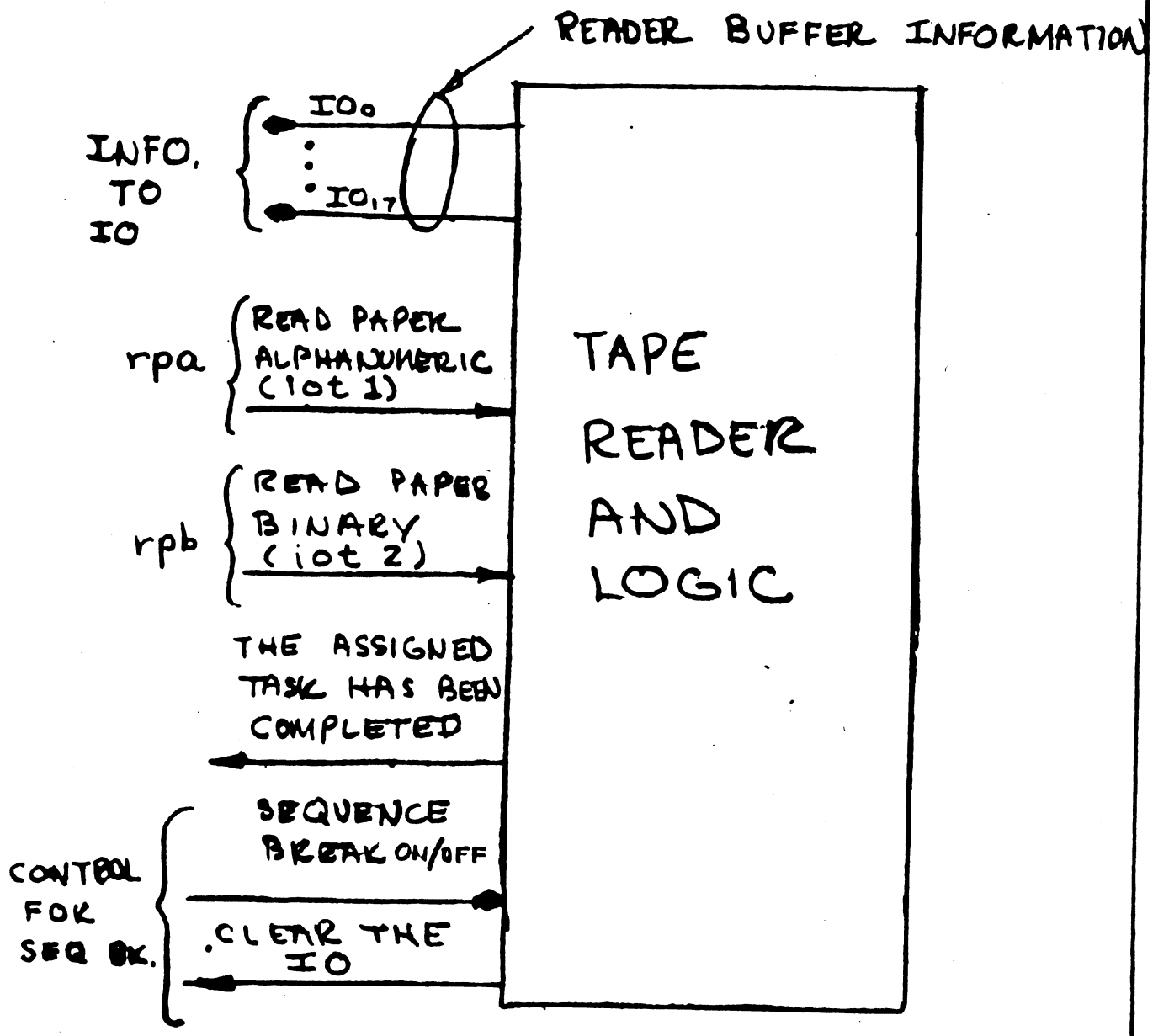
The Photoelectric Tape Reader

The logic for the photoelectric tape reader is shown in figure 9. Five, 6, 7, or 8 hole tapes are read at a 400 line per second rate. The paper tape reader has an 18 bit buffer which holds the information that is gathered from tape. There are two modes of operation, read paper alphanumeric (rpa) and read paper binary (rpb). When the command rap 720001 or 730001 is given, one line of tape is read. All eight holes of the line go into the right 8 bits of a buffer. When the 8 bits are assembled in the buffer, a pulse appears on the line clear the In-Out register, then 2.5 microseconds later the completion pulse reads the reader buffer information into the IO.

The command read paper binary, or rpb = 720002 or 730002, etc, reads three lines of paper tape into the reader buffer and then returns a completion pulse. For a line of tape to be recognized in this mode, the eighth hole must be punched while the seventh is ignored. The information is packed in the buffer as three 6-bit characters.

If the sequence break system is "on" the completion pulse still occurs, but the In-Out register is not cleared prior to the completion pulse. In this case, the reader buffer contents may be transfered to the IO by the command read reader buffer, rrb = iot 31. The sequence break system will be discussed below.

Computation can proceed during the 2.5 or 7.5 milliseconds an rpa or rpb is being carried out. Synchronization must be handled as previously described.



400 LINE PER SECOND TAPE READER

FIGURE 10

THE GENERAL CONNECTION OF PROGRAMMED IN-OUT TRANSFERS

Outgoing Information

Lines are available from the In-Out register output for transfer of information to a device. The individual lines of each bit of a register form a bus to which connections may be made. Appendix I describes the loading restrictions of these lines.

The In-Out register may connect with output devices. The scheme for affecting this connection to the output bus is shown in figure 11. Here, each bit of the In-Out register, i.e., the one's side output is available at a taper pin connector block, being suitably buffered with a bus driver. This taper pin block allows connections to be made in parallel to devices which desire information.

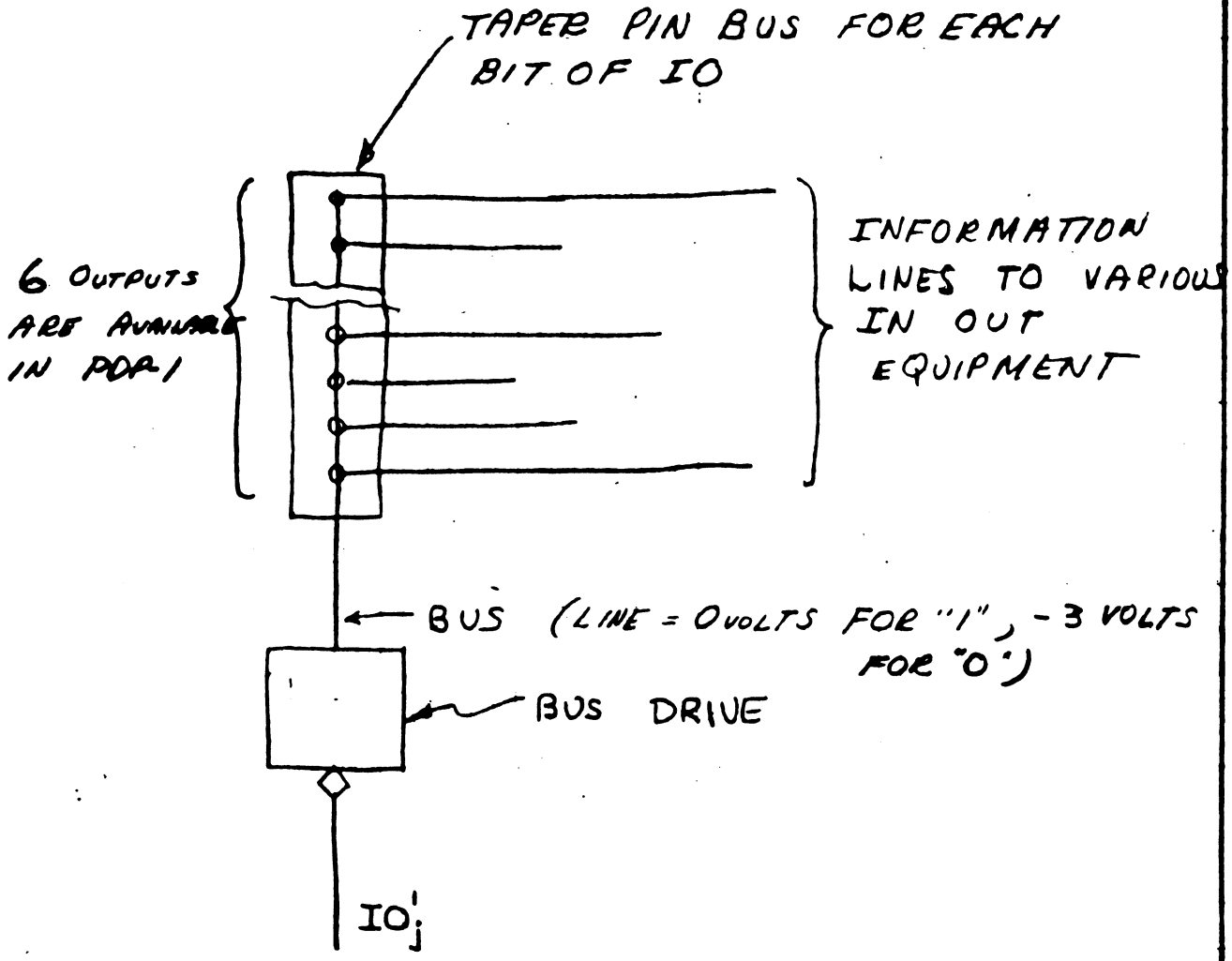
Incoming Information

A similar taper pin block arrangement is available for connecting the outputs of a device to the inputs of the In-Out register mixer. The In-Out Mixer reads in the various information bits to the In-Out register. The input levels "and" with program in out transfer pulses to form a pulse which is mixed with similar pulses through an "OR" circuit. One bit of the In-Out register Input Mixer is shown in figure 5. Eight taper pin inputs exist for several devices. The specifications for the input mixer are summarized in Appendix I.

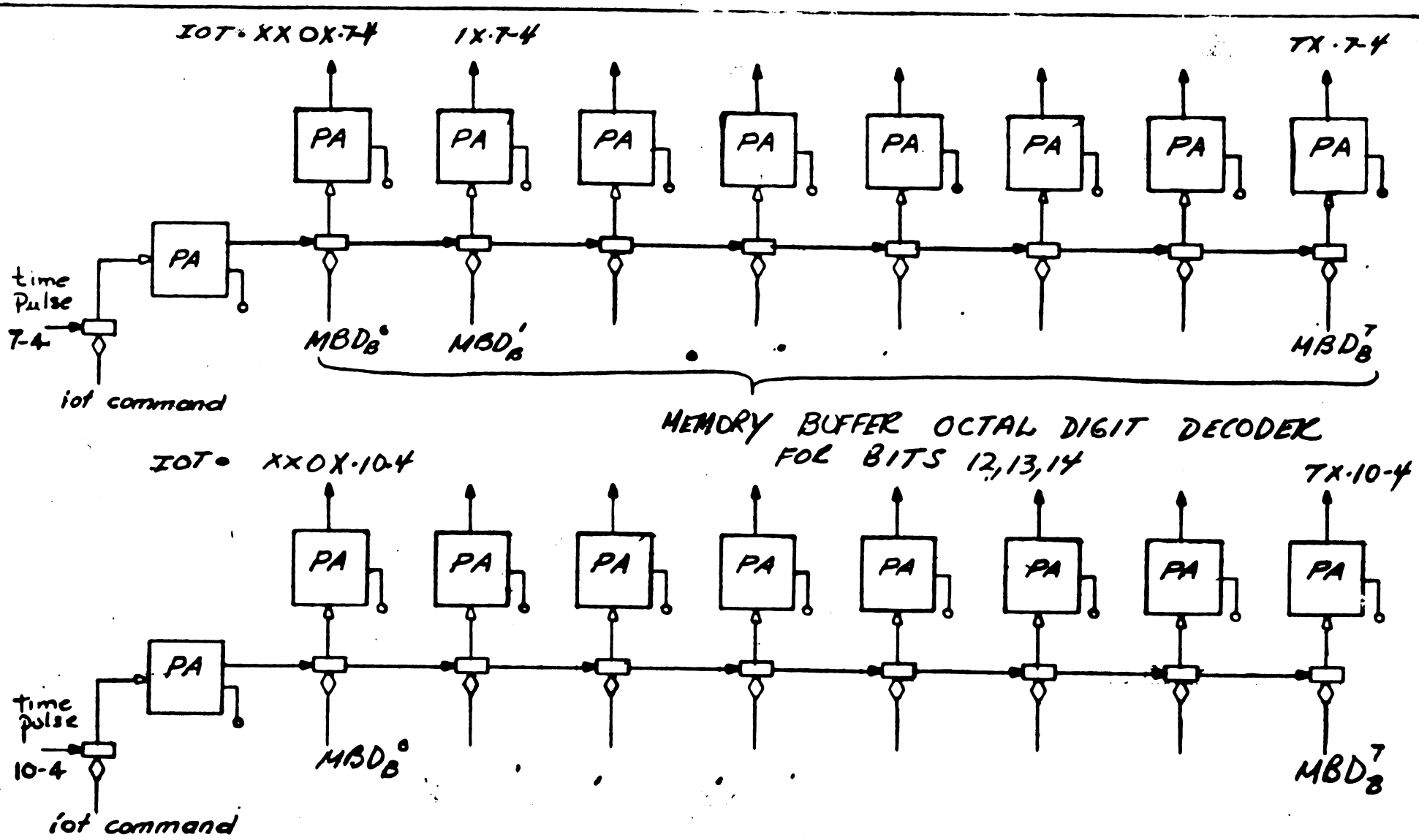
Formation of Programmed In-Out Transfer Pulses

A portion of the In-Out Transfer Control block schematic is shown in figure 12. This diagram shows the formation of the standard In-Out transfer pulses, (or in some cases pairs of pulses). These include rpa, rpb, tyo, tyi, etc., i.e., all those pulses discussed above which communicate with the conventional equipment.

Appendix I describes the Memory Buffer Lines which constitute the basic decoding for the in-out transfer pulses. In general, the levels are not needed externally. The lines, MBD with subscripts, A, B, C, and D, and superscripts 0 - 7, refer to the decoding of the Memory Buffer octal digits, MBD, or Memory Buffer Decoders. Decoder A is connected to bits 15 - 17, decoder B to 12 - 14, etc. The superscript number refers to the decoder output lines.

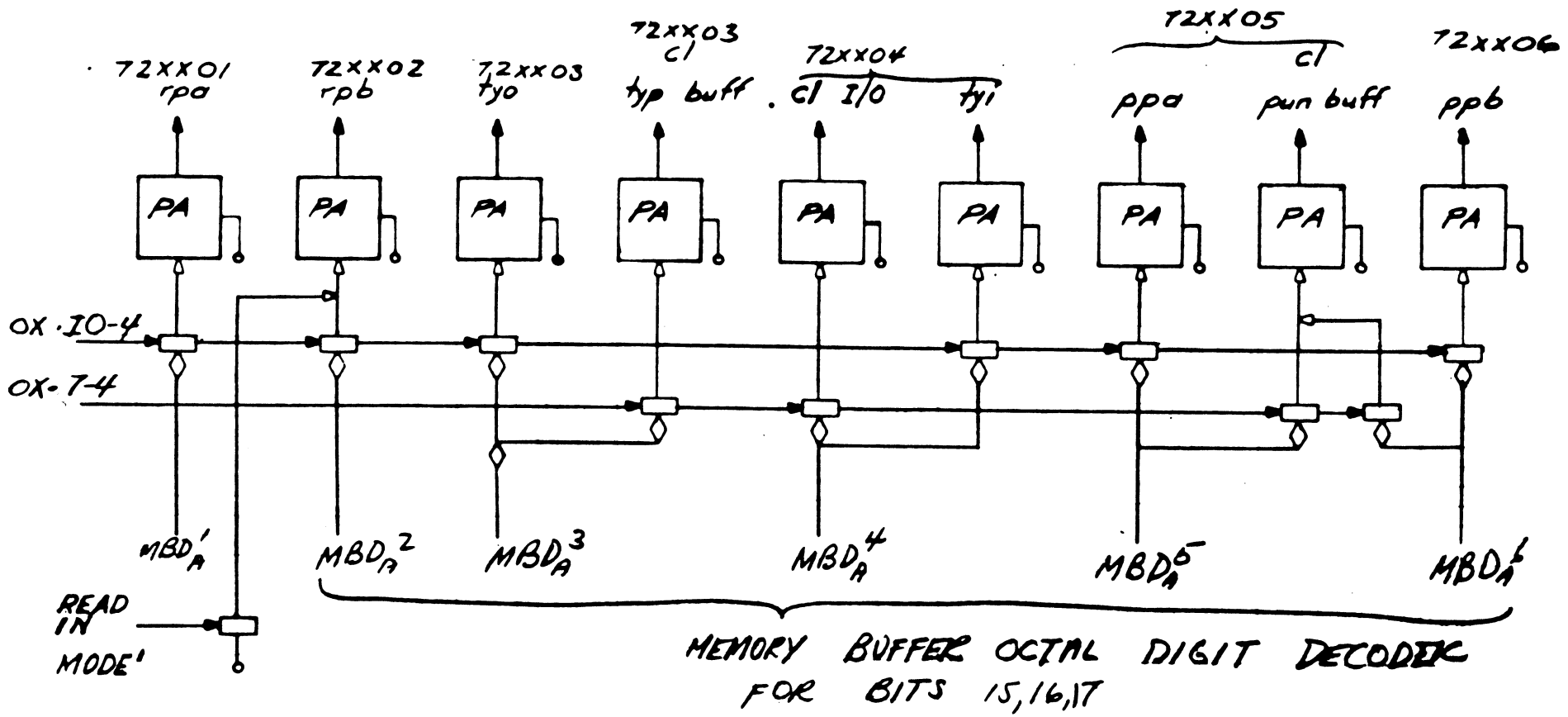


IN OUT REGISTER
OUTPUT BUS
FIGURE 11



FIRST STAGE IOT ADDRESSING

FIGURE 12a



FINAL STAGE IOT ADDRESSING

FIGURE 12b

Two pulse lines are used to form basic in-out transfer pulses. They are called tp7-4 and tp10-4. These line pulses are fixed relative to the computer's memory cycle time, with tp7-4 and tp10-4 occuring 2.5 and 5.0 microseconds after the beginning of a memory cycle. The -4 indicates that the pulse is a DEC 0.4 microsecond pulse. The iot instruction level is shown and is a one when an In-Out transfer command is given.

The pulse formed by "anding" tp7-4 (or tp10-4) with the In-Out transfer command is again "anded" with MBD_B of figure 12a to form 8 basic pairs of pulses.

The 8 by 8 decoding scheme allows up to 64 addressable pulses (or pairs) to be easily formed. Any or all of these pulses may be further subdivided by using MBD_C or MBD_D. Thus, a particular iot may be connected to an additional 8 by 8 array to give size to 64 sub-orders. If the physical space limitations are somehow circumvented, 4096 distinct in-out instructions can be implemented. A list of the in-out transfer commands is given in Appendix II.

The iot control pulse amplifier labeled "read paper alpha" emits a pulse when Memory Buffer Decoder A is a 1, ($MBD1_A$, the 3 least significant bits in the In-Out transfer instruction are 001), and when MBD_B (or the 3 next least significant bits are 000) and when an iot command is given.

Synchronization of Device Completion and Computer Restarting

The computer must synchronize with some in-out devices. This synchronization is determined by the use of bits 5 and 6, in the iot command (see page 8). The implementation of this function is shown in figures 13a and b. A flip-flop is used as a switch for each device which is to operate in this mode. Thus, the various return pulses (completion pulses) may or may not affect the restart action.

Restart synchronization need not use the flip-flop switch arrangement. In this case, a restart pulse can go directly to the in-out transfer done, pulse amplifier to restart the machine.

The action of bit 5 to indicate a wait words on any iot instruction. The enable/disable feature of the completion pulse

is only applicable to devices having the extra logic of figure 13a. The standard devices (reader, punch, and typewriter output) have this feature. The oscilliscope display has this feature.

If the Sequence Break System is included, the device completion pulse may also go to an appropriate SBS channel.

Certain iot commands will never use the wait feature and thus the completion enable/disable need not be included. In such a case, bit 6 of the command may be used in the same manner as bits 7-11.

Special Levels and Pulses

Appendix I describes several connections which are useful for input-output equipment.

Accumulator Outputs: The Accumulator bits 0-11 outputs are available. The driving power of the AC is limited and loadings should be carefully observed.

Memory Buffer and Octal Memory Buffer Decoders: The various Memory Buffer Octal digit levels and Memory Buffer levels are available. When iot commands are given, these may be used either for information or additional decoding.

Program Flags: An external device can set a program flag. Either of two inputs may be used. The output of the program flag is available for control purposes.

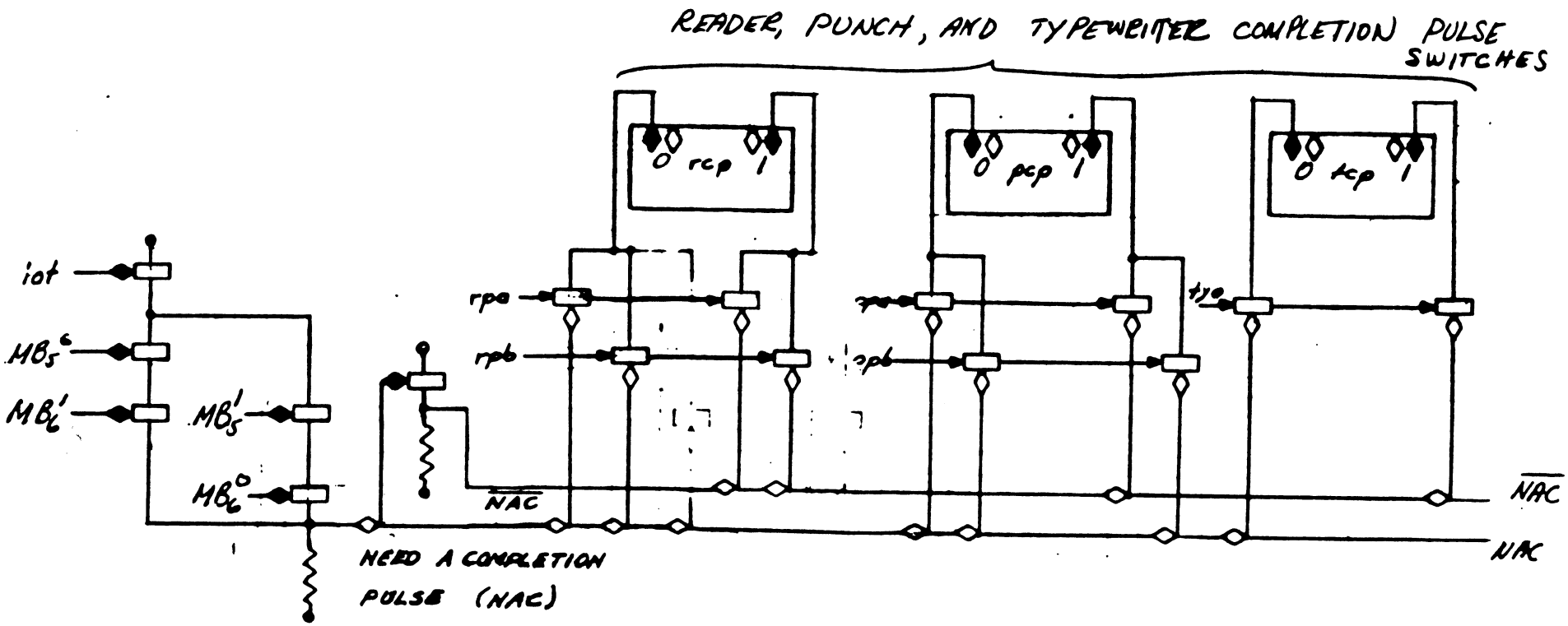
Program Counter: A pulse can add one to the program counter. In this way, an instruction could be skipped if an in-out condition is not met.

Synchronization Pulses: Timing pulses are available which may be used to synchronize extra equipment. The time pulses occur at 1,2.5 and 5.0 microseconds after the start of a memory cycle. The start and stop pulses are available and occur when the console start or stop keys are pressed.

BASIC SEQUENCE BREAK SYSTEM

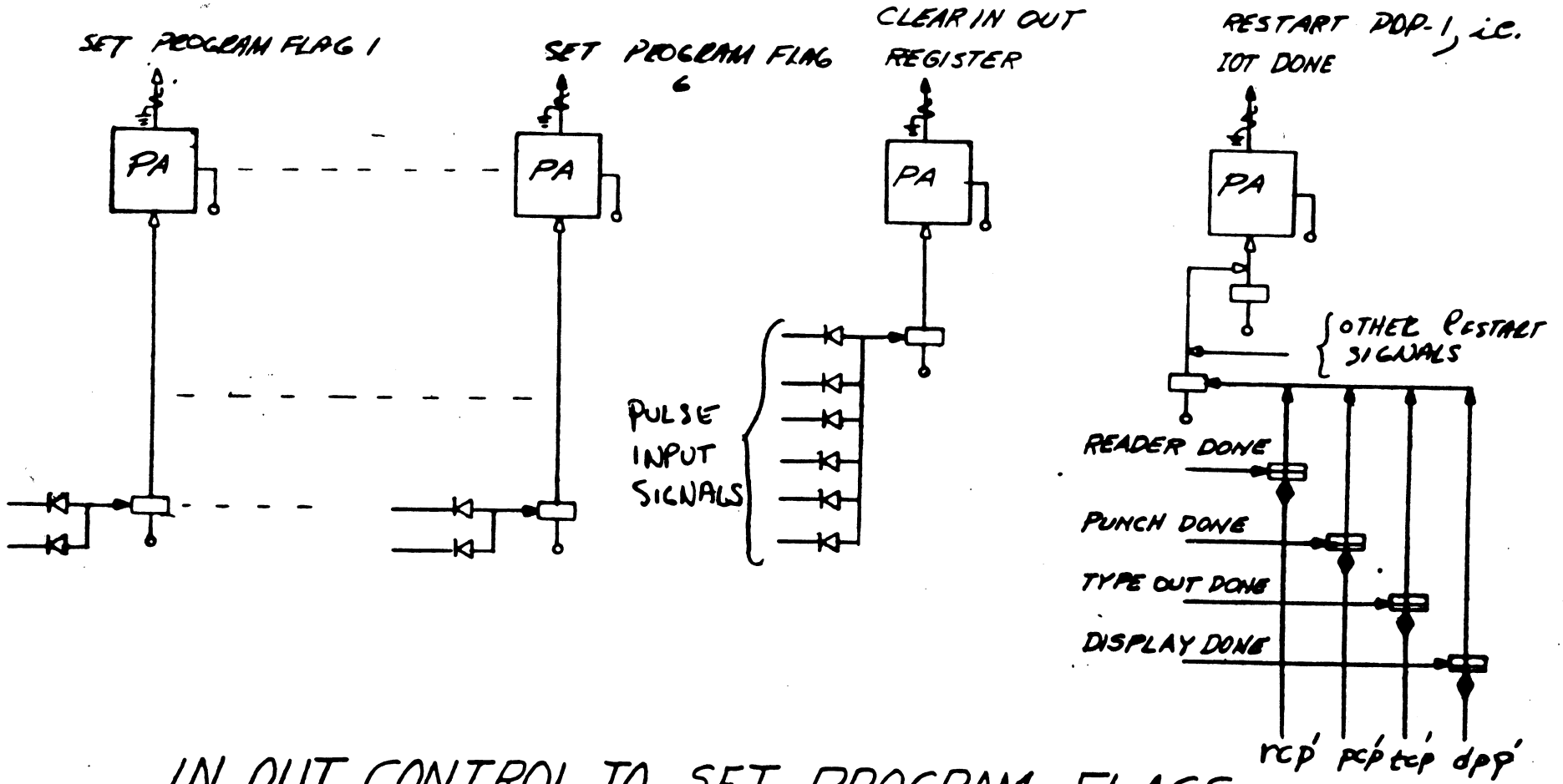
Programmed In-Out Synchronization

When a device communicates in a random fashion with PDP-1, a



CONTROL SWITCHES FOR COMPLETION PULSES

FIGURE 13a



IN OUT CONTROL TO SET PROGRAM FLAGS, CLEAR IN OUT REGISTER & RESTART COMPUTER

FIGURE 13b

means is required to sample the device's readiness, then to synchronize information transfers. The program flags may be set by external devices and sensed by program. If the machine must always "wait" for device completion (or action), before continuing a sequence, periodic sampling of the program flags, or any other external states (e.g., flip-flops) must be affected.

On page 12, a program loop was given which continually checked the typewriter input flag (program flag 1). When a character has been typed, a program sequence can continue, handling the character. The limitations of this technique are:

1. Operating time is consumed sensing and waiting for flag.

2. Priorities among devices must be established by the program. This often requires careful considerations.

3. A program loop (loop contains one sense flag instruction) can be no longer than the repetition period of a device.

Automatic Program Interruption

The Sequence Break System (SBS) enables a program sequence to be interrupted, i.e., the sequence of instructions broken, each time a device needs attention. Thus, a program loop need not be used to sense a condition, but rather calculations may proceed, and are only interrupted when communications need exist (e.g., data transfers) between the machine and a device.

The basic Sequence Break System has provisions for 12-level inputs. Thus, when any of the 12 levels are present, (a one) an interruption occurs. This program interruption takes the following form:

1. The contents of the AC are stored in register 0.

2. The contents of the program counter (and memory field information) are stored in register 1.

3. The contents of IO are stored in register 2.

4. The program resumes in register 3.

The above steps require 3 x 5 microseconds. Since one of 12 devices may cause the interruption, a program must find the device. The command, check status bits, cks, 72 0032, causes a set of status bits to be read into the IO. By examining the various IO bits, the appropriate device may be found. While status bits 0-4 are taken with conventional in out equipment, the remaining bits may be free for use.

When an interruption occurs, the action of other devices connected to other SBS inputs are ignored, until the break is terminated properly. This is accomplished by restoring the C(AC), C(IO) and returning to the previous sequence. This termination may be affected by the following program:

```
lac 0
lio 2
jmp * 1
```

The last instruction, 61 0001, is the one which actually affects the termination, and must be given exactly as shown above. Two additional instructions turn on or turn off the sequence break mode.

1. Enter sequence break mode, esm, iot 55.

2. Leave sequence break mode, lsm, iot 54.

When the sequence break mode is off, no interruptions may occur.

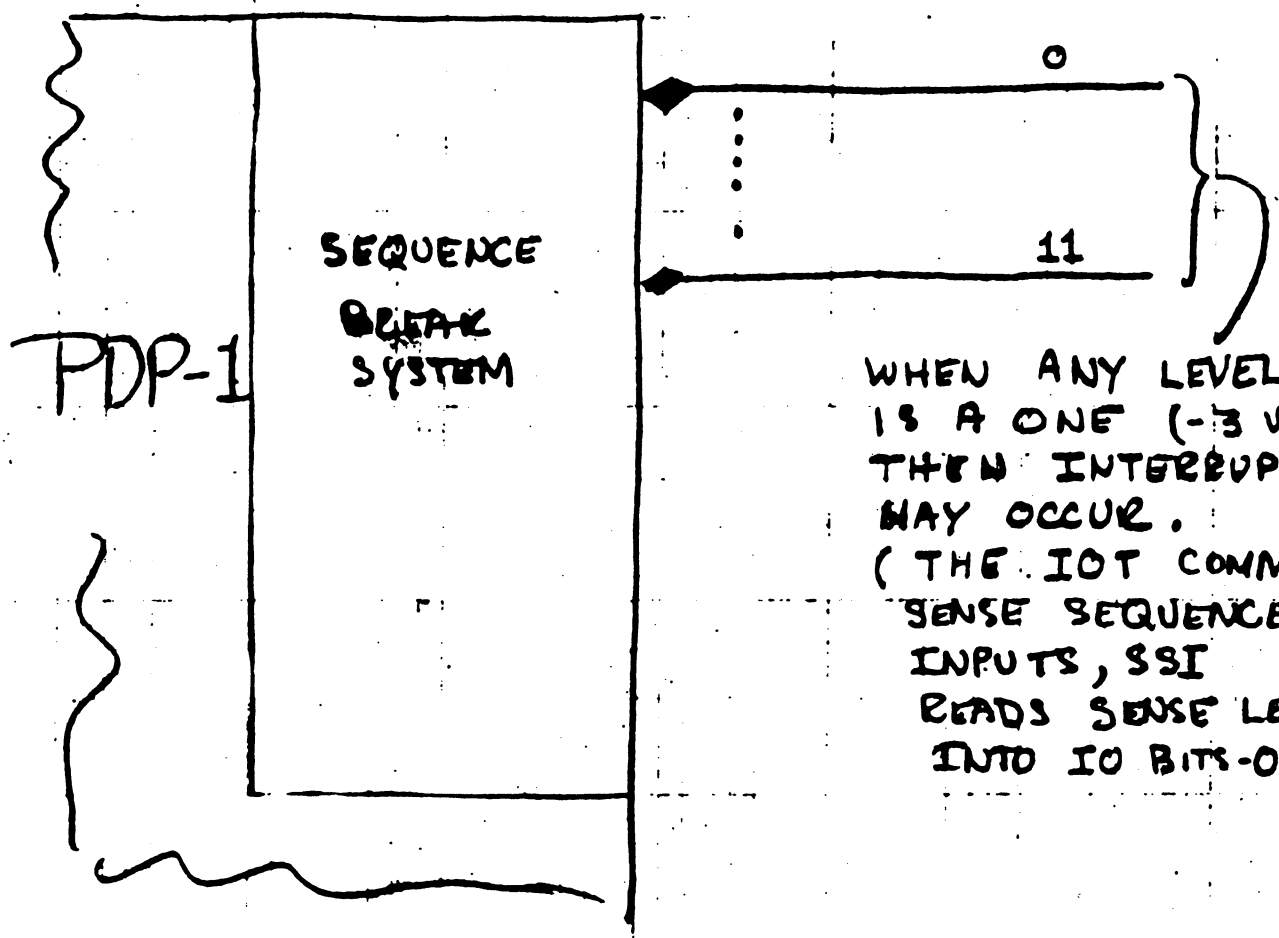
The sequence break mode can be turned on or off by the console "start" switch, either by being pushed up for on or down for off. A block diagram of the sequence break inputs is shown in figure 14.

The program flags may be wired to act as inputs to the SBS System.

TYPE 20 - 16 CHANNEL SEQUENCE BREAK SYSTEM

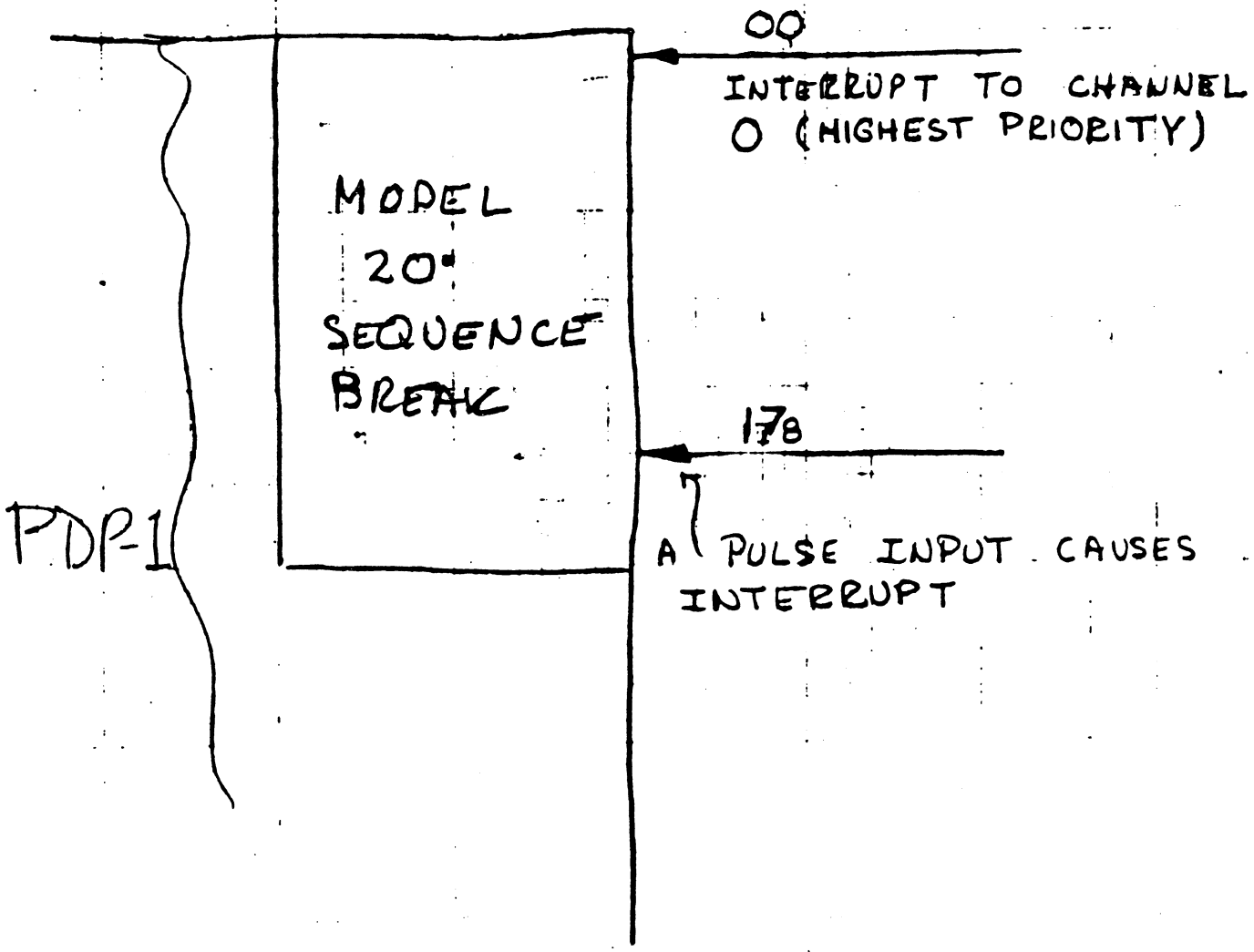
This system allows 16 devices to operate simultaneously, and in a preassigned priority. The 16 separate channels allow a device to interrupt to 1 of 16 unique locations (instead of one for the basic break system). Thus, 16 x 4 memory locations are assigned to this sequence break. The channels are arranged in a priority chain.

The block diagram for the system is given in figure 15. In



CONVENTIONAL SEQUENCE BREAK SYSTEM

FIGURE 14



MODEL 20 - 16 CHANNEL SEQUENCE BREAK SYSTEM

FIGURE 15

this case, a pulse or a negative transition can cause a break to a sequence. Program commands may cause these pulses to be ignored, i.e., a channel may be "on" or "off". Since the 16 channels are in a priority chain, a higher priority break may interrupt the sequence of a lower one. The following action occurs if a pulse enters the ith channel input:

1. C(AC) = C(4i)
2. C(PC) = C(4i + 1)
3. C(IO) = C(4i + 2)
4. Program resumes in 4i +3 by a jmp command.
   where i = 0, 1, ... 17_8

The break is terminated by the following:

```
lac 4i
lio 4i + 2
jmp * 4i + 1
```

The termination is affected by the last instruction.

Flip-Flops of Type 20 Sequence Break

There are 16 x 4 flip-flops which have significance and are visible to the user. The flip-flops are:

1. Channel ON
2. Break Synchronize
3. Waiting Break
4. Break Started

Channel On is used as a switch and must be in the "one" state to allow device completion pulses to set Break Synchronize.

Waiting Break and Break Synchronize act together. Waiting Break is set to a one when Break Synchronize is a one. Two microseconds after this event, Break Synchronize is cleared.

Waiting Break provides a signal that a particular channel would like an interruption. This interruption may occur provided that a break has not started (Break Started = 1) for the same or a higher channel and no higher channel is currently Waiting Break.

When a break occurs to a channel, Break Started is set to a one. At this time, Waiting Break is cleared. Higher priority channels may interrupt lower channels. Break Started remains a one until the break is dismissed by giving a jmp * instruction as previously discussed.

The instructions for Model 20 Sequence Break Systems are:

1. esm, Enter Sequence Break System mode, iot 55, (see basic sequence break).

2. lsm, Leave Sequence Break mode, iot 54. (See basic sequence break).

3. isb, Initiate Sequence Break: Channel i is selected by bits 8-11 of the command, iot 52. This command allows the program to initiate a break, or simulate an external pulse action. This break is not conditioned by "channel on".

4. asc, Activate Sequence Break: Channel i is selected by bits 8-11 of the command iot 51. Allows channel i to be active, i.e., pulses to effect breaks).

5. dsc, Deactivate Sequence Break, iot 50. Channel i is selected by bits 8-11 of the command, iot 50. A channel is turned off thus any incoming break pulses will be ignored.

HIGH SPEED CHANNELS FOR DIRECT MEMORY DATA TRANSFERS CONCURRENT WITH COMPUTATIONS

A high speed channels feature is an option which may be used to transfer blocks of words between memory and an in-out device, e.g., a device such as magnetic tape, which is time consuming to program on a character by character (or word by word) basis.

This optional feature is installed with tape control, Type 52 or may be installed separately for special applications. When the feature is added, the internal machine facilities for three channels are available. The three channels are serviced on a demand basis.

A common information transfer requirement is that successive words go to or from the computer at high speed. The use of program flags, the sequence break, or a program loop of in-out transfers, all effect the speed and efficiency of the transfers. In most cases,

the number of program steps required to transfer each word is small with respect to the number of program steps over the whole period between transfers. For high speed devices, such as magnetic tape, the reduction in available time between transfers makes it desirable to reduce the transfer sequence time and improve the ratio of available computation time to total running time.

Each of the three channels are interrogated regularly at the completion of each memory cycle, on a priority basis. The priority is wired and fixed. The sequence break system has a priority just below the last high speed channel. If a channel has a word for memory, or requires a word from memory, a memory cycle interruption occurs. The diagram of one high speed channel (of 3) is shown in figure 16.

The lines operate as follows:

1. The memory address and field lines select the memory register for the device.

2. Information

   a. Memory buffer outputs contain the information which is to be transmitted to a device.

   b. The high speed channel input mixer is similar to the in-out register input mixer and is used for incoming information.

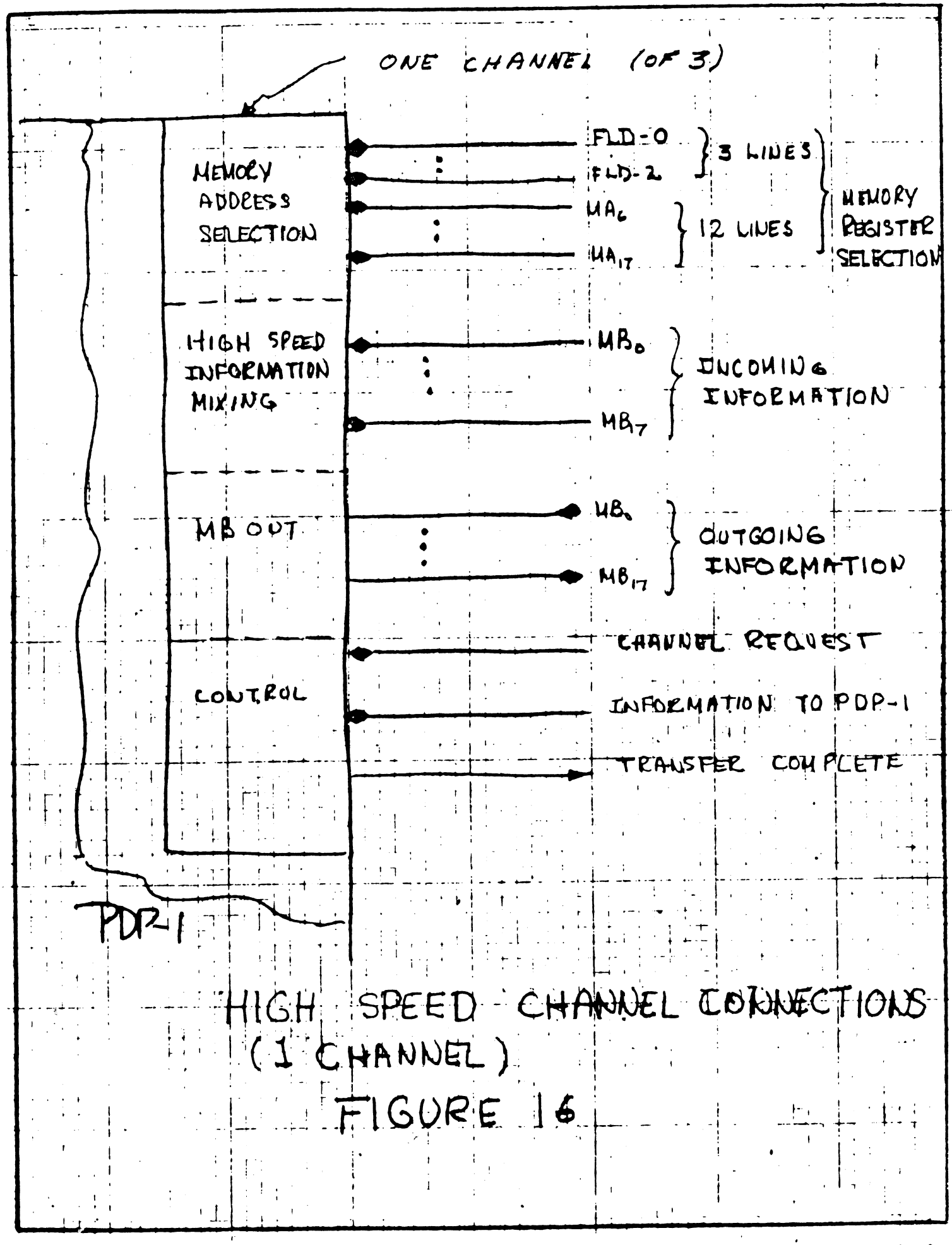
3. Control

   a. Request - When the line is a one, a memory interruption may take place.

   b. Transfer Direction Lines - This line specifies whether information is to go to or from PDP-1.

   c. Transfer Done - A pulse occurs on this line to acknowledge the request completion. The request line must be removed to prevent another interruption.

The timing diagram for a high speed channel is shown in figure 17.



HIGH SPEED CHANNEL CONNECTIONS
(1 CHANNEL)

FIGURE 16

APPENDIX I

LIST OF IN OUT TRANSFER COMMANDS

Instruction	Code	Definition
Photoelectric Punched Tape Reader		
rpa	iot** X001	Read punched tape, alphanumeric forward direction
rpr	iot X101	Read punched tape, reverse direction
rpb	iot XX02	Read punched tape, bi-octal
rrb	iot XX30	Read reader buffer (for sequence break operations)
cks	iot XX33	Check status bits. Miscellaneous control bits are read into IO. Bit 0 - Displayed point seen by light pen 1 - Paper tape reader busy 2 - Typewriter busy 3 - Typewriter key struck 4 - Punch busy
Tape Punch		
ppa	iot XX05	Punch punched tape, alphanumeric
ppb	iot XX06	Punch punched tape, bi-octal
Typewriter		
tyo	iot XX03	Type out
tyi	iot XX04	Type in

*Symbolic assembly mnemonic code.
**iot has the value 720000 in symbolic assembly language.
X denotes octal digit may be anything.

Instruction	Code	Definition
Display - Type 30		
dpy	iot XX07	Display one point on CRT
(The above commands, excepting rpr, are standard for all PDP-1's)		
Card Reader - Type 41-523		
rac	iot XX41	Read a card
rsc	iot XX42	Row synchronize and clear field counter
raf	iot XX32	Read a field of 18 columns and index field counter
Card Punch - Type 40-523		
pac	iot XX43	Punch a card
psc	iot XX44	Row synchronize and prepare to punch first field
pag	iot XX22	Punch a field of 18 columns and index field counter
Basic Magnetic Tape - Type 51		
mcb	iot XX70	Magnetic tape clear buffer
mwc	iot XX71	Magnetic tape write character
mrc	iot XX72	Magnetic tape read character
mcs	iot XX34	Magnetic tape check status
msm	iot XX73	Magnetic tape select mode
High Speed Magnetic Tape - Type 52		
muf	iot kk75	Magnetic tape unit and final address, kk specifies control information

Instruction	Code	Definition
mic	iot kk76	Magnetic tape initial address and command
mel	iot kk35	Magnetic tape examine location
mes	iot kk36	Magnetic tape examine status
mri	iot kk66	Magnetic tape reset initial address
mrf	iot kk67	Magnetic tape reset final address
Clock		
rsk	iot XX47	Reset the clock
rdk	iot XX37	Read clock time into IO
Timer		
stm	iot XX24	Set timer with IO
Relay Buffer		
srb	iot XX21	Set relay buffer
Analog to Digital Converter		
cnv	iot XX41	Convert a voltage
rcb	iot XX31	Read converter buffer
Sequence Break System - Type 20		
esm	iot XX55	Enter sequence break mode
lsm	iot XX54	Leave sequence break mode
asc	iot NN51	Activate sequence break channel NN

Instruction	Code	Definition
dsc	iot NN50	Deactivate sequence break channel NN
isb	iot NN52	Initiate sequence break to channel NN
Core Memory Expansion		
cfd	iot kk74	Change fields, Type 11 expansion kk specifies new field
cdf	iot Xk74	Change data field, Type 14 expansion k specifies new data field
Line Printer - Type 62		
lgo	iot X145	Line printer go (print)
lfb	iot X245	Fill line printer buffer
lsp	iot X345	Space (vertically) line printer

APPENDIX II - LIST OF INTERCONNECTIONS WITH PDP-1

LINE	IN-OUT	NUMBER	POLARITY	DEC CIRCUIT AT PDP-1	SUGGESTED LOAD	REMARKS
IO_{0-17}	out	7	◆— = 1	1685	100 Type 4128 etc. cap. diode gates - Less than 1000 ft. cable	General programmed output transfers
AC_0	out	1	◆— = 1	4113	cap. diode gates	For oscilloscope, but available
AC_{1-11}	out	1	◆— = 1	4113	cap. diode gates	For oscilloscope, but available
MB_{0-17}	out	3	◆— = 1	1685	cap. diode gates	For In Out and High Speed Channel Transfers
$MBD_A \cdot iot$	out	8	→	4603		Used only for forming in out transfers (2 sets available, 2.5 and 5.0 microseconds after start of cycle).
MBD_B	out	8	◆— = 1	4113		Used only for forming in out transfers
$MBD_{C,D}$	out	8	◆— = 1	4113	cap. diode gates	For iot commands, sequence break
program flag	out	1 x 6	◆— = 1	1209		For external level output
IO_{0-17}	In	1	◆— = 1	4129		For incoming information, levels must be present 2.0 microseconds (4 wired and available with additional 9 - 4129)
MB_{0-17}	In	3	◆— = 1	4129		For incoming information of High Speed Channels
Memory fields						
MA_{6-17}	In	3	◆— = 1	4129		To select register for High Speed channels
to IO Restart	In	6	→	4110		Clear IO
Computer	In	6	→	4110		Restarts computer when in out halting
program flags	In	2 x 6	→	4112		Any device may set flag

APPENDIX II - LIST OF INTERCONNECTIONS WITH PDP-1

Page 2

LINE	IN-OUT	NUMBER	POLARITY	DEC CIRCUIT AT PDP-1	SUGGESTED LOAD	REMARKS
1 PC Basic Sequence	In	6	→	4110		Advance program counter
Break	In	12	—◆ = 1	4110		Any level causes Sequence Break
Time Pul. 2.4	Out	1	→	4603		Pulses must be buffered by 4603 before driving separate lines
						occurs 1 microsecond after start of Memory Cycle
7.4	Out	1	→	4603		occurs 2.5 microseconds after start of Memory Cycle
10.4	Out	1	→	4603		occurs 5.0 microseconds after start of Memory Cycle
Power on	Out	1	→	4603		occurs when power comes on
stop	Out	1	→	4603		occurs when console stop button is pushed
start	Out	1	→	4603		occurs when console start button is pushed
Sequence Break Type 20	In	16	→	4128		Pulse causes Sequence Break, Type 20 only
High Sp. Channel requests	In	1 x 3	—◆ = 1	4106		Requests a channel transfer
High Sp. Channels In/Out	In	1 x 3	—◆ = 1	4106		Direction of a channel transfer
High Sp. Channels Complete	Out	1 x 3	→	4603		Specifies completion of transfer

When a break occurs to a channel, Break Started is set to a one. At this time, Waiting Break is cleared. Higher priority channels may interrupt lower channels. Break Started remains a one until the break is dismissed by giving a jmp * instruction as previously discussed.

The instructions for Model 20 Sequence Break Systems are:

1. esm, Enter Sequence Break System mode, iot 55, (see basic sequence break).

2. lsm, Leave Sequence Break mode, iot 54. (See basic sequence break).

3. isb, Initiate Sequence Break: Channel i is selected by bits 8-11 of the command, iot 52. This command allows the program to initiate a break, or simulate an external pulse action. This break is not conditioned by "channel on".

4. asc, Activate Sequence Break: Channel i is selected by bits 8-11 of the command iot 51. Allows channel i to be active, i.e., pulses to effect breaks).

5. dsc, Deactivate Sequence Break, iot 50. Channel i is selected by bits 8-11 of the command, iot 50. A channel is turned off thus any incoming break pulses will be ignored.

HIGH SPEED CHANNELS FOR DIRECT MEMORY DATA TRANSFERS CONCURRENT WITH COMPUTATIONS

A high speed channels feature is an option which may be used to transfer blocks of words between memory and an in-out device, e.g., a device such as magnetic tape, which is time consuming to program on a character by character (or word by word) basis.

This optional feature is installed with tape control, Type 52 or may be installed separately for special applications. When the feature is added, the internal machine facilities for three channels are available. The three channels are serviced on a demand basis.

A common information transfer requirement is that successive words go to or from the computer at high speed. The use of program flags, the sequence break, or a program loop of in-out transfers, all effect the speed and efficiency of the transfers. In most cases,