PROGRAMMED DATA PROCESSOR-6 HANDBOOK

PDP-6

PROGRAMMED DATA
PROCESSOR - 6
HANDBOOK

Figure 1 Typical PDP-6 System

PROGRAMMED DATA PROCESSOR — 6

36-bit word length ■ 15 index registers and/or accumulators
FORTRAN II — MACRO-6 assembler — utility programming library
Integrated hardware and software for time sharing ■ Microtape
Asynchronous operation, modular construction ■ Memory overlap
Core memories up to 262,144 words, 2 μsec, directly addressable
Fast memory 16 words, 0.4 microsecond ■ 128 input-output devices
363 instructions ■ fast floating point — multiply 14 μsec average
Program assignable operation codes ■ Byte manipulation, half word
Block transmission ■ Seven channel priority interrupt system
Programmed input-output transfers require no data channels
Multiple processors ■ Remote input-output ■ Mass memory

CONTENTS

CHAPTER 1
PDP-6 SYSTEM DESCRIPTION

CHAPTER 2
ARITHMETIC PROCESSOR TYPE 166

CHAPTER 3
MEMORY

CHAPTER 4
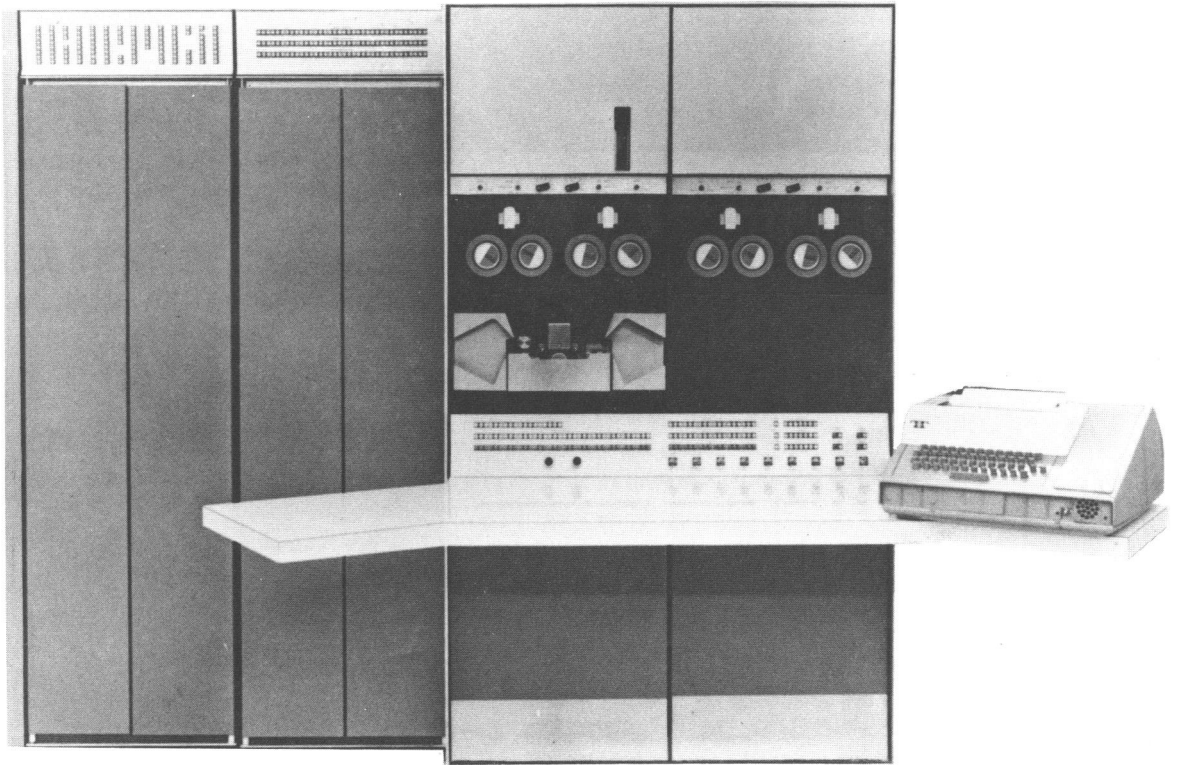## INPUT-OUTPUT

ILLUSTRATIONS

Figure 2 Arithmetic Processor Type 166

CHAPTER 1
PDP-6 SYSTEM DESCRIPTION

Programmed Data Processor-6 (PDP-6) is a general-purpose digital computing system designed for scientific data processing. The flexibility of this system permits the user to specify the data handling capacity and the exact configuration needed to meet his requirements. The system can be expanded with presently available equipment or, at a later date, with equipment yet to be developed. Faster memories, for example, can be added as they become available.

PDP-6 design eliminates the need for off-line conversion equipment. Conversion of programs from cards or paper tape to magnetic tape can be done concurrent with normal program running. Users at peripheral teleprinters can simultaneously prepare and debug their programs on line.

The PDP-6 system consists of processors, memories, and input/output devices. Since each is autonomous (no device is dependent upon another for its timing), a system configuration can include memory modules of different speeds, processors of different types sharing the same memory modules, and standard or unique input/output devices.

For maximum flexibility of system configurations, the PDP-6 system is built around two busses: processor-memory bus and processor-input/output bus. The memory bus permits each processor to directly address 262,144 words of core memory, automatically permits overlapping, and simplifies multiprocessor operation. An input/output bus of a processor can service up to 128 devices.

The operating system consists of a supervisory control program, system programs, and system subroutines. Included are a Symbolic Assembler and Macro Processor, a FORTRAN II Compiler, and debugging aids. A library of general utility programs is also provided.

Neither the processors nor any of the standard peripheral equipment require an air-conditioned environment or floor reinforcement. Ordinary 115-volt power is sufficient for all equipment.

Processors

A PDP-6 system can include any number of processors of the same or different types. The Type 166 is a 36-bit arithmetic processor with many powerful features, including 16 accumulators, 15 index registers, built-in floating point arithmetic, and byte operations capability. Memory protection and relocation registers are included for time-sharing operations.

The Type 167 I/O Processor gives direct memory access to high speed devices, such as drums, discs, and displays. It takes over local control of data transfers after receiving system commands and initial conditions from the arithmetic processor. Thereafter the two processors operate asynchronously, so that I/O transfers are carried out in parallel with arithmetic processing.

Up to three controls, such as the Type 236 Drum Control, can be connected to the Type 167 I/O Processor.

INPUT / OUTPUT BUS

MEMORY MODULE

ARITHMETIC PROCESSOR 166

PERFORATED TAPE READER 760 — 400 CPS READER

PERFORATED TAPE PUNCH 761 — 63.3 CPS PUNCH

TELE-PRINTER CONTROL 626 — CONSOLE TELE-PRINTER

CARD READER 461 — 200/800 CPM READER

CARD PUNCH 460 — 100/300 CPM PUNCH

LINE PRINTER 646C — 1000 LPM PRINTER

DATA COMM. SYSTEM 630 — TTY LINE / TTY LINE — UP TO 64

DISPLAY CONTROL AND MONITOR 346 — MONITOR 343 / MONITOR 343 — UP TO 4

MAGNETIC TAPE CONTROL 516 — TRANS 570 / TRANS 570 — UP TO 8

DATA CONTROL 136

MICROTAPE CONTROL 551 — TRANS 555 / TRANS 555 — UP TO 8

MEMORY MODULE

MEMORY MODULE

I/O PROCESSOR 167

DRUM CONTROL 236 — DRUM / DRUM / DRUM / DRUM

MEMORY MODULE

MEMORY MODULE

TO OTHER PROCESSORS

Figure 3 PDP-6 System Interconnections

PDP-6 — A MODULAR SYSTEM

The diagram shows the three main parts of a PDP-6 system, memories, processors, and input/output equipment, interconnected with busses. Memories and input/output devices contain their own buffer registers and control circuits, including decoders to recognize commands from the processors. The Data Control Type 136 is a double-buffered device which is used not only with magnetic tape equipment, as shown here, but with any high speed, block transfer device, such as drum or disc.

The system shown in this diagram is a theoretical one of very high capacity, but it is entirely within the capability of PDP-6. Memory size, indicated by the modules on the left, can be as large as 262,144 words per processor, and up to four processors can address a given memory module. Very high speed devices, such as drum, tape, disc, and display, can have direct access to the memory system through the I/O Processor Type 167. The combination of the asynchronous nature of the system and the characteristics of the memory bus makes possible truly simultaneous memory references by two or more processors. Sequential memory references made by one processor can be overlapped.

PDP-6 is also a highly effective system in a minimum configuration. All system programs will operate in a system consisting of a Type 166 Arithmetic Processor, a memory module of 16,384 words, a Microtape system, and a teleprinter. Later expansion of either the memory or input/output system can be made with no change whatever in the existing system. Memory modules can be of any speed: A low cost system might call initially for slower (5-microsecond) memories, later to be augmented by faster memories (down to 0.5-microsecond).

PDP-6 systems are thus completely adaptable to current and future requirements, both technical and budgetary.

Input/Output

The input/output bus consists of device selection, data, control, and status sense lines. A 7-channel program-assignable priority interrupt system signals the processor when input/output devices require service. Word count and memory address registers are located in the processor and are available to all devices. This reduces the cost of various input/output controls, and permits data block transfers between tapes, card readers, printers, displays, and other devices.

Memory

The PDP-6 core memory subsystem permits modular expansion using blocks of different sizes and speeds. The types 163B and 163C core memory modules contain 8,192 and 16,384 words, respectively. Each has a word length of 36 bits, a cycle time of 2 microseconds, and an access time of 0.8 microseconds. The Type 162 Fast Memory Module contains 16 words with a 0.4-microsecond cycle. Slower core memories, such as the 5-microsecond Type 161, can be used where economy is an overriding criterion.

The memory-processor bus permits memory cycle overlap, gives all processors direct access to memory, and permits easy expansion and modification of the memory subsystem. In addition, the bus allows the processors to remain connected to memory only as long as needed to transfer information: That is, a processor can put a word on the bus and resume operations as soon as the memory acknowledges, without waiting for the memory to store the word. Similarly, when reading a word out of memory, the processor takes the information and operates on it immediately, without waiting for the memory to finish the rewrite portion of its cycle.

Maximum system efficiency is achieved when sequential memory references address alternate memory modules. The addressed module places data on the bus as soon as it is available in the memory buffer and disconnects itself from the bus while rewriting, freeing the processor to store the result or seek the next instruction in a second memory module before the first one has completed rewriting. Utilizing such overlapping memory references, PDP-6 users can effectively cut in half the time required for average random accesses. Multiple connections between the bus and each memory module permit module sharing on a priority basis for multi-processor operations.

PROGRAMMING SYSTEM

The programming system for PDP-6 consists of a supervisory control program, system programs, and library routines. The entire system is designed to run on any PDP-6 system with at least 16,384 words of core memory, a console teleprinter, and a Microtape system. However, the programming system is modular. Parts of it can run on machines with smaller memory capacities. For example, programs can be assembled with MACRO6 using the above-mentioned input/output equipment and only 8,192 words of core memory.

A variety of programs are provided through the Digital program library, and a continuous in-house program design effort regularly improves and expands the library.

The Supervisory Control Program

This is the name given to a collection of programs remaining permanently in memory to provide overall coordination and control of the total operating system. The segments of the program are:

COMMAND CONTROL PROGRAM handles all commands addressed to the system from the user-consoles. These commands would include requests to log in or out, a request to use the edit program, requests to have a program placed on the run queue, requests to load a program, etc.

PROGRAM SCHEDULER is called at regular intervals to decide which program in memory is to be run. A running program is temporarily terminated each time its allotted time has run out or when it requires input/output operations with a device that is busy. A program may be terminated temporarily by user intervention to the scheduler, or it may suspend its own operation. Temporary termination does not remove the program from memory. A program may be dumped on backing storage and permanently discontinued by calling the scheduler and allocator.

FACILITIES ALLOCATOR is called any time an I/O device or memory space is required. It may be called from a user-console or by a running program. Under this program one user-console is designated the operator console. As such it has special facilities available which are not available to other consoles, such as line printer assignments. Storage is permanently assigned for all resident programs, that is, those programs that are in memory at all times. Finally, "logical" tape assignments are made. Two Microtape units are designated the system library and the system scratch tape. Two other tapes may be assigned as peripheral input tape, used to prepare jobs to be stacked from cards or paper tape, and system input tape, used to input a full tape. (Each user-console may require a Microtape unless programs requiring files are to be run.)

COMMAND DECODER preprocesses commands from the user-console. This program is used to convert parameters, etc., before the command is sent to the program for which the command is intended.

I/O CONTROL is called whenever an I/O device is to be used. This program assigns equipment, controls the I/O devices, controls data transfers between memory and the I/O device, and controls the buffering of data for the device. (Users provide the necessary buffering for devices which require file buffering.) All program I/O instructions are trapped, and the actual control of the I/O operation then passes to the I/O Control Program.

System Programs

These are the programs designed to implement system functions which may be requested from the user-console. This is in contrast to system subroutines which may be called by system programs or other programs. System programs are normally provided by Digital, but they may be provided by each installation for its users. The programs contain a mode by which they may be terminated to return the communication link to the system. Some of the system programs are described below.

EDITOR PROGRAM manipulates the text of a named file on a Microtape or in the user area of the drum (corresponding to Microtape). This file may be used for the creation of text or for later use as data or as a program to be translated by the FORTRAN Compiler, etc. The commands provided for the editor allow text to be created, deleted, or moved about.

PERIPHERAL CONVERSION PROGRAM handles all those jobs normally done by a separate peripheral processor. The priority interrupt system and multiple memory accumulators in the PDP-6 eliminate virtually all loss in running time. Such processing is done through the arithmetic processor.

INTER-CONSOLE MESSAGE PROGRAM switches message traffic between the various user-consoles. This program allows the user to request manual operations by the operator and receive acknowledgment. Such an operation would be the mounting/dismounting of user tapes.

LINKING LOADER PROGRAM accepts programs in a form produced by the translators, and produces an area of core memory loaded with the program. Upon request, it may also produce a storage map of the loaded programs along with symbol tables. Several programs may be linked together in loading. The loader requests special library tapes to be loaded, and verifies that the program has been completely loaded.

TRANSLATOR DISPATCHER is called to load the FORTRAN, MACRO6, or other translators. The translators are rather large programs that do not reside in memory, but are stored on the System Library tape until they are called into memory by the translator.

FORTRAN II COMPILER accepts FORTRAN II input statements and produces relocatable binary output coding for later loading by the Linking Loader. Compiling is done in one pass. PDP-6 FORTRAN II is an extension of the conventional FORTRAN II language to give the user more facilities and to take advantage of PDP-6 hardware. The ASCII character set is used. Subscripts may consist of statements (fixed or floating). Any number of dimensions may be used to specify an array. Signed integers have 36-bit values, but when used as subscripts are truncated to 18 bits.

MACRO6 ASSEMBLY PROGRAM translates MACRO6 input language to a relocatable binary output for the Linking Loader. MACRO6 is a two-pass assembly program, and the language provides for macro instruction definitions and usage. Literal assignments are made by brackets [ ]. Numbers may be expressed as binary, octal, decimal, and floating point. Text may be placed in a binary program by the occurrence of the "text" data generating statement, and "byte" causes a string of bytes to be assigned and packed into a word. The "repeat" control statement causes the statements following the control to be repeated "n" times.

DEBUGGING PROGRAM (DDT) is loaded with a program and allows all assembly language programs to be debugged. The program may be started or stopped, words in the program may be modified, and DDT may search the program looking for particular words. DDT may also be used in a "trace" or break point mode, and the program is run until a particular location (a break point) is encountered.

THE SYSTEM SUBROUTINES include:
1. I/O Format Control which provides for the various format statements used in the FORTRAN II language. These subroutines are also available to other programs and may be called from the systems library tape.

2. Arithmetic Subroutines which include all the arithmetic subroutines required for FORTRAN II, such as, sine, cosine, $\log_E$, $\log_{10}$, exponent, tangent, arctangent, and square root.

INPUT/OUTPUT EQUIPMENT

Digital offers a large selection of optional equipment for full utilization of the extensive input/output capacity of the system.

MICROTAPE TRANSPORT TYPE 555
A fixed-address magnetic-tape facility for high speed loading, readout, and on-line program debugging. Read, write, or search speed is 80 inches a second. Density is 375 bits an inch. Total storage is three million bits. Features phase recording, rather than amplitude recording; redundant, nonadjacent data tracks; and a pre-recorded timing and mark track.

MICROTAPE CONTROL TYPE 551
Controls up to eight Type 555 Microtape Transports. Searches in either direction for specified block numbers, then reads or writes data. Uses the Type 136 Data Control to assemble data and buffer transfers to the processor.

DATA CONTROL TYPE 136
Provides for assembly of 6, 12, 18, or 36-bit characters; six input/output devices can be controlled.

TELEPRINTER AND CONTROL TYPE 626
Permits on-line programming and debugging. Provides hardcopy outputs. Is standard Teletype equipment, operating at ten characters a second.

TELEPRINTER INTERFACE TYPE 630
Automatically scans up to 64 teleprinter (TTY) lines. Signals a program interrupt when teleprinter needs service.

CARD PUNCH CONTROL TYPE 460
Permits on-line punching of cards in any format, including IBM, at 100 or 300 cards a minute.

CARD READER AND CONTROL TYPE 461
Provides on-line reading of standard punched cards at 200 or 800 cards a minute in alphanumeric or binary codes.

HIGH SPEED PERFORATED TAPE PUNCH AND CONTROL TYPE 761
Punches 8-hole tape at 63.3 characters a second.

HIGH SPEED PERFORATED TAPE READER AND CONTROL TYPE 760
Reads perforated paper tape photoelectrically at 400 characters a second.

MAGNETIC TAPE CONTROL TYPE 516
Automatically controls up to eight tape transports Type 570 or IBM 729 series. Permits reading, writing, forward/backward spacing, rewind and unload, and rewind. Uses a Type 136 Data Control to assemble data and buffer transfers to the processor. Longitudinal and lateral parity are checked.

MAGNETIC TAPE TRANSPORT TYPE 570
Tape motion is controlled by pneumatic capstans and brakes, eliminating conventional pinch rollers, clamps, and mechanical arms. Tape speed is either 75 or 112.5 inches a second. Track density, program-selectable, is 200, 556, and 800

bits per inch. Tape width is one-half inch, with six data tracks and one for parity. Format is compatible with IBM NRZI. Dual heads permit read checking while writing.

I/O PROCESSOR TYPE 167

Establishes a data transmission path between main memory and the drum(s). Up to four drums can be connected to the I/O processor.

MAGNETIC DRUM AND CONTROL TYPE 236

Drum stores 1,048,576 36-bit words organized into 128 tracks, each with 8192 words consisting of 64 128-word blocks. A word is transferred in 6.4 microseconds, and the drum revolution time is 52 milliseconds.

DISPLAY CONTROL AND MONITOR TYPE 346

Plots points, lines, vectors, and characters on a 9⅜-inch-square raster of 1024 points along each axis. Time between points plotted is 1.5 microseconds in the vector, increment, and character modes. In random point plotting, a time of about 35 microseconds is required per point.

DISPLAY MONITOR TYPE 343

Provides additional cathode ray tube display for multiple consoles.

HIGH SPEED LIGHT PEN TYPE 370

Detects data displayed by the Types 346 and 343 and inputs identifying signal to the computer.

AUTOMATIC LINE PRINTER AND CONTROL TYPE 646C

Prints 1000 lines a minute, 120 columns a line, any one of 64 characters a column.

AUTOMATIC LINE PRINTER AND CONTROL TYPE 646A

Prints 300 lines a minute, 120 columns a line, any one of 64 characters a column.

ANALOG-TO-DIGITAL CONVERTER TYPE 138

Transforms an analog voltage to a binary number, selectable from 6 to 11 bits. Conversion time varies, depending on the number of bits and the accuracy required. Twenty-one combinations of switching point accuracy and number of bits can be selected on the front panel.

MULTIPLEXED ANALOG-TO-DIGITAL CONVERTER TYPE 138/139

The Type 139 Multiplexer Control permits up to 64 channels of analog information to be applied singly to the input of the Type 138 Analog-to-Digital Converter. Channels can be selected in sequence or by individual addresses.

HIGH-SPEED ANALOG-TO-DIGITAL CONVERTER TYPE 142

Transforms an analog voltage to a signed, 10-bit binary number in 6 microseconds. Conversion accuracy is $\pm0.15\%$ $\pm1/2$ least significant bit.

ANALOG-DIGITAL-ANALOG CONVERTER SYSTEM TYPE ADA-1

Performs fast, real-time, data conversion between digital and analog computers. Maximum sample rate for D/A conversion is 200 kc; for A/D and interlaced conversions, 100 kc. Digital word length is 10 bits. Actual conversion times are 5 microseconds for A/D and 2 microseconds for D/A. Semiautomatic features enable the converter system to perform many of the functions that a computer normally performs for other converter interfaces.

CHAPTER 2
ARITHMETIC PROCESSOR TYPE 166

The Type 166 is a general purpose processor which performs all of the arithmetic and logical operations in a PDP-6 system and transfers data to and from input-output devices. It is connected to the memory and input-output subsystems through busses and bus interfaces.

The logic block diagram, Figure 4, shows the interconnection of the principal registers of the Type 166, and their functions are described below. An unusual feature of the 166 is that when appropriate the contents of the AR and/or the MQ are transmitted to memory at the end of each instruction. Thus the state of the machine, except for the PC, the four flags, the memory protection and relocation registers, and the executive mode status, is constantly in memory, and there are no instructions that directly reference processor registers except the PC and the flags. Using the AR for the effective address calculation will never destroy a partial result in a calculation. Further, the processor is free to carry out any operation, for example, an I/O operation called for by an interrupt.

PROCESSOR REGISTERS

MA	(Memory Address) An 18-bit register that specifies the desired location to the memory system.
MB	(Memory Buffer) A 36-bit data buffer that communicates with the memory system. The MB is also used as the addend register in arithmetic operations and contains one of the operands in logical operations.
AR	(Arithmetic Register) A 36-bit register used in indexed address calculations, as an operand register for logical instructions, and in all arithmetic and shifting instructions. The accumulator specified by the instruction is moved to the AR during most instructions.
MQ	(Multiplier Quotient) A 36-bit register used as an extension of the AR in operations requiring 72 bits.
MI	(Memory Indicator) A 36-bit register used to display the contents of memory registers. The MI register is set equal to the contents of the memory location specified by the ADDRESS switches each time the processor references the location.
PC	(Program Counter) An 18-bit register that contains the memory location from which the next instruction is to be taken.
IR	(Instruction Register) An 18-bit register that contains the instruction code, accumulator number, indirect bit, and index register number for the instruction being executed.
SC	(Shift Counter) A 9-bit register that contains the number of times a particular part of an instruction is to be repeated. This applies to the shifting, floating point, multiply, divide, and byte manipulation instructions.
FE	(Floating Exponent) A 9-bit register used to store temporary results during floating calculations.

MP   (Memory Protection) An 8-bit register used by the executive system
            in the user mode. This register contains the largest address that can
            occur in a user's program, that is, the size of the block of memory
            allocated to the user. When making a reference to memory, the MP
            and MA registers are compared. If the MA is larger than the MP, the
            memory cycle does not take place, the memory protection flag is
            set, and control goes to an executive program with the computer in
            the executive mode.

RL   (Relocation Register) An 8-bit register used by the executive system
            in the user mode. If the address in the MA does not exceed the largest
            address specified by the MP, and is greater than 17_8, the contents of
            the RL register are added to the high order 8-bits of the MA before
            the memory cycle takes place.

Figure 4   Data Flow Diagram

In addition to the registers shown in Figure 4, there are several flags important to programming.

AR CRY0   Arithmetic register Carry 0. This flag is set when there is a carry from bit 0 of the arithmetic register.

AR CRY1   Arithmetic register Carry 1. This flag is set when there is a carry from bit 1 of the arithmetic register.

AR OV   Arithmetic register Overflow flag. This flag is set when the sign of a result is not consistent with the signs of the operands.

PC CHANGE   Program Counter Change. This flag is set whenever control is transferred by a skip or jump instruction, except for a JFCL instruction with bit 12 set to 1.

PDL OV   Push Down List Overflow. A designated accumulator is incremented or decremented by 1000001_8 during the execution of the push down instructions. This flag is set if a carry from bit 0 occurs.

BIS   Byte Increment Suppression. This flag, when set to 1, suppresses incrementing during a byte instruction that calls for incrementing. This flag is necessary since a byte instruction may be interrupted between incrementing and the byte operation. In such a case, the incrementing must be suppressed during the first execution of the instruction following the interrupt. The flag is automatically set if such an interrupt occurs. It is cleared before the byte instruction is completed.

COUNT CLOCK ENABLE   This flag is set by a CONO instruction. If set, it enables the Clock flag to cause an interrupt. (See Processor I/O instructions.)

CLOCK FLAG   This flag is set periodically by a clock. If the Enable flag is set, and if a priority channel is assigned to the processor flags, it causes an interrupt.

PC CHANGE ENABLE   This flag is set by a CONO instruction. If set, and a priority channel has been assigned to the processor flags, it enables the PC Change flag to cause an interrupt.

OV FLAG ENABLE   This flag is set by a CONO instruction. If set, and if a priority channel has been assigned to the processor flags, it enables the AR OV flag to cause an interrupt.

EXEC MODE   This mode flip-flop is set when the executive program is running. In the executive mode, the program has access to all of memory. In the user mode (executive mode flip-flop is reset) the protection and relocation registers specify the area of core in which programs may be run. In the user mode illegal memory references and instructions which use the I/O equipment or attempt to halt the computer set the executive mode flip-flop.
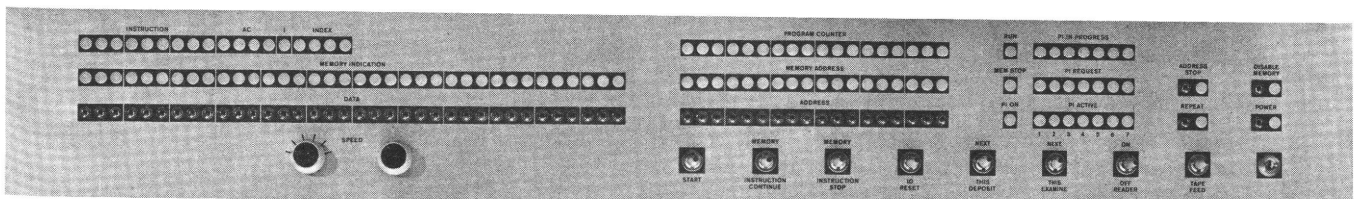

Figure 5   The Control Panel

CONSOLE CONTROLS AND INDICATORS

Lever Switches	Function
START	Depressing this switch resets the program counter to the address contained in the ADDRESS switches and starts the processor.
	Raising the switch does the same as the above except the machine is started in shadow mode. In the shadow mode, all references to memory locations $0-17_8$ go to the core memory rather than to fast memory. The shadow mode is terminated whenever an instruction located above 17_8 is executed. Therefore, if the address in the ADDRESS switches is greater than 17_8, the effect of the switch is the same whether it is raised or depressed.
	The switch has no effect if the machine is running.
STOP	Stops the processor at the end of the memory or instruction cycle. When the switch is in MEMORY STOP position, the read-pause-write mode of operation is inhibited; the machine stops at the end of both the read cycle and write cycle.
CONTINUE	If the processor has stopped at the end of an instruction or memory cycle, the program may be resumed by depressing this key. The combined use of the STOP and CONTINUE keys has the effect of a single step key.
I/O RESET	Clears the I/O System.
DEPOSIT	In the THIS position, the contents of the 36 DATA switches are deposited in the contents of the memory location specified by the ADDRESS switches. The machine will not be stopped if it is running.
	In the NEXT position, the memory address register is incremented by one, and the contents of the 36 DATA switches are deposited in the memory location specified by the contents of the memory address register. This key position has no effect if the machine is running.
EXAMINE	In the THIS position, the contents of the memory register addressed by the ADDRESS switches are displayed in the memory indicator lights. If the machine is running, it is not stopped.
	In the NEXT position, the memory address register is incremented by one, and the contents of the memory location specified by the memory address register are displayed in the memory indicator lights. This key position has no effect if the machine is running.
READER ON-OFF	Turns the power to the perforated tape reader on or off.
TAPE FEED	Feeds blank tape through the tape punch.

Toggle Switches	Function
CONSOLE DISABLE	This is a key-locked switch that disables all control switches that can interfere with the operation of the machine.

ADDRESS STOP	Stops the processor each time the contents of the memory address register are equal to the contents of the ADDRESS switches. An adjacent indicator light is on when the switch is on. The existence of an address stop condition inhibits the read-pause-write mode of memory operation, and the machine stops at the end of both the read cycle and write cycle.
REPEAT	If on while a lever switch is held down, the effect is as if the lever switch were being pressed repeatedly with a frequency determined by the setting of the speed knob and vernier.
SPEED	Two controls that vary the repeat interval when the REPEAT switch is on. The left knob is a six-position coarse control giving speed ranges from 3.4 microseconds and increasing by a factor of ten up to 340 milliseconds. The right knob is a continuously variable fine control.
POWER	Turns on power to the processor and all equipment connected to it.
MEMORY DISABLE	This switch inhibits the non-existent memory continue pulse. If a non-existent area of memory is addressed, the non-existent memory flag is set; and if the MEMORY DISABLE switch is on, the machine stops.
DATA	Up to 36 bits of information can be inserted for transfer to memory when the DEPOSIT switch is pressed. (May be examined under program control. See I/O Programming for Type 166 Processor.)
ADDRESS	There are 18 address switches which are used with various keys and switches.

Indicator Lights	Function
INSTRUCTION	Displays bits 0-8 of the instruction register.
AC	Displays bits 9-12 of the instruction register.
I	Displays bit 13 of the instruction register.
INDEX	Displays bits 14-17 of the instruction register.
MEMORY INDICATION	Displays the contents of the memory register addressed by the ADDRESS switches. The MI is set whenever the EXAMINE key is operated or the program references the register.
PROGRAM COUNTER	Displays the contents of the program counter. When the processor stops, the program counter will be pointing to the next instruction.
MEMORY ADDRESS	Display the last location referenced in memory before the machine stopped. If the machine stopped on a halt instruction (JRST 4), the MA contains the address of the halt instruction.
RUN	Lights when the machine is running.
MEM STOP	Lights and machine stops when end of memory cycle is reached if STOP switch is on or if an ADDRESS STOP occurs.
PI ON	Lights when priority interrupt system is enabled.

Indicator Lights	Function
PI ACTIVE	Display which PI channels are enabled.
PI REQUEST	Display which channels are requesting an interrupt.
PI IN PROGRESS	Display which channel is actually causing an interrupt.

PROGRAMMING FOR TYPE 166 PROCESSOR

For the Type 166 the first 16 locations of memory (0–17$_8$) function as accumulators, index registers, and ordinary memory locations. The state of the system resides entirely in memory except for the program counter and the AR flags. There are, therefore, no instructions which refer explicitly to the arithmetic elements of the processor and no need to protect partial results in the event of an I/O operation.

The function to be performed by one of these 16 locations is determined by the instruction word field in which the address appears. The processor always addresses memory in exactly the same way regardless of how it intends to use the data. Consequently, from the point of view of the memory system there is nothing unusual about the first 16 locations except that they may reside in a fast memory module, a desirable though not necessary option. Depending on the field in which the address appears, the processor uses the data received as an accumulator operand, as an index register for the effective address calculation, or as an ordinary memory operand. Thus, there is no need for separate groups of instructions to perform accumulator, index register, and memory manipulations. It should be noted that only the right halves of the 15 memory locations 1–17$_8$ are used as index registers.

Some instructions use two accumulators: the accumulator specified by the accumulator field of the instruction word, and the one following. The accumulator following 17$_8$ is taken to be accumulator 0.

Some areas of memory are reserved for special functions and must be used with caution. They are:

20$_8$–37$_8$ Reserved for loader if fast memory is not available for the shadow mode.

40$_8$–41$_8$ Reserved for unused operation code traps.

42$_8$–57$_8$ Program interrupt system.

It is advisable that memory locations up to 100$_8$ be left unused to allow room for future expansion.

There are five major categories of Type 166 Processor instructions: data transmission, arithmetic and logical operations, executive instructions, I/O instructions, and push down instructions (a hybrid group consisting of both data transmission and control instructions).

The data transmission group may be divided into four subgroups: full word transmission; half word transmission (particularly useful for address and index manipulations); byte manipulation (particularly useful for storing and retrieving information in segments of less than one memory word); and two miscellaneous instructions, an exchange instruction and a block transmission instruction.

The arithmetic and logical operations may also be divided into four subgroups: fixed point arithmetic, floating point arithmetic, Boolean instruction, and shift instruction.

There are five subclasses of executive instructions: memory or accumulator modification and testing, a group of instructions that will increment or decrement an accumulator or memory word by one and/or test to determine a jump or skip; arithmetic compare, a group that will compare a number in an accumulator with a number in memory (fixed or floating point) and skip as specified; logical and modify compare, a group that will test and/or modify masked bits of an accumulator and skip as specified; jump instructions; and some miscellaneous instructions.

The push down and I/O groups are both very small groups with four and eight instructions, respectively.

MODES — Some instruction classes consist of basic instructions and modes. Groups of instructions which move and/or combine information require a transmission mode. Transmission modes specify the direction in which the information is to be moved, e.g., memory to accumulator or accumulator to memory; they may specify where to leave the results of an operation, e.g., in memory, in an accumulator, or both; or they may specify what parts of the result from an operation are to be stored, e.g., low order bits and/or remainders in floating point operations. Finally, there is an immediate mode which directs the instruction to take its effective address as the memory operand.

Groups of instructions which compare and skip or jump require modes to specify the conditions under which the skip or jump is to occur.

UNUSED OPERATION CODES — There are 26 unused operation codes reserved for future expansion. They have no effect on the machine and may be thought of as no-operation instructions.

NO OPERATION INSTRUCTIONS — Some combinations of instructions and their modes have no effect on the state of the machine. These may be considered as no-operation instructions.

PROGRAMMED OPERATORS — The 64 instruction codes with zeros in the first three bits have been reserved for programmed operators. When used, bits 0–17 of the instruction are stored in bits 0–17 of location 40. The result of the effective address calculation is stored in bits 18–35 of location 40. The instruction in location 41 is then executed. Presumably, this instruction would be a jump to a routine that would interpret the programmed operator. It should be a jump which stores the program counter in order to return to the main program.

Instruction Word Format

There are two formats for the Type 166 Processor instructions, the basic instruction format and the I/O instruction format. The basic format is shown in Figure 6.
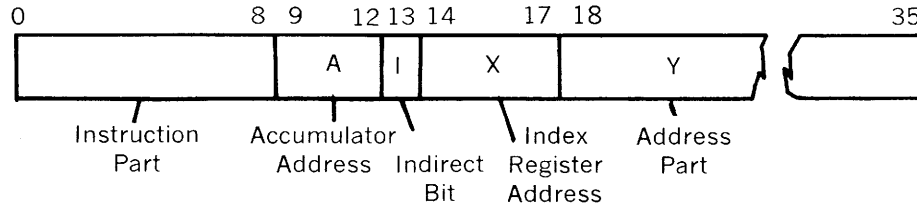
Figure 6 — Basic Instruction Format

All instructions, with the single exception of the I/O, use this basic format. The functions of I, X, and Y, which are common to the basic and I/O formats, are described below under Effective Address Calculation. Bits 0-8 specify the instruction code, including the mode. Bits 9-12 specify the accumulator to be used by the instruction. The format for the I/O instruction is shown in Figure 7.
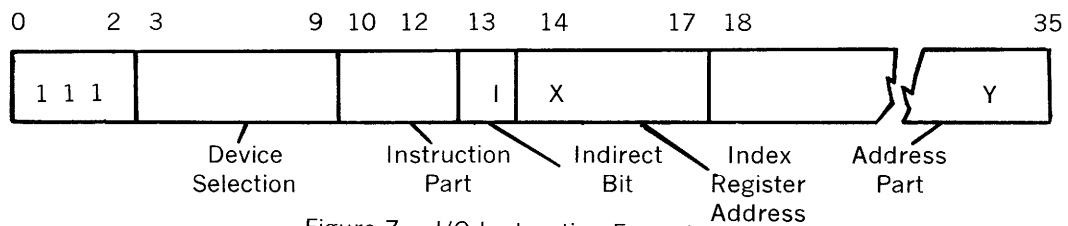
Figure 7 — I/O Instruction Format

Bits 3-9 are used to specify any one of the 128 possible I/O devices, where the processor itself and the program-interrupt system are considered to be the first of the devices (device #000 000 0 and #000 000 1 respectively for use with the conditions-out and conditions-in instructions). Bits 10-12 are used to specify any one of the eight possible I/O instructions.

EFFECTIVE ADDRESS CALCULATION

All instructions in the PDP-6, without exception, calculate an effective address using bits 13-35 in exactly the same way. The steps are:

1. Obtain the number in the address field Y, bits 18–35. Any one of 262,144 locations can be specified.

2. If the index field X, bits 14–17, is non-zero, then add the contents of the specified index register to the number obtained in step 1.

3. Obtain the indirect bit, I, bit 13. If it is 0, the calculation is done and the result of steps 1 and 2 is the effective address. If it is 1, then go to step 4.

4. Use the address calculated by steps 1 and 2 to obtain a new word from memory, and go back to step 1.

The effective address calculation continues until a word is encountered with a 0 in bit 13. At that point the result of steps 1 and 2 is taken as the effective address for the instruction.

This calculation is carried out for all instructions. When the immediate mode is specified, if one or more indirect addresses are involved, the effective address is not the number appearing in the address field of the word. Rather, it is the first result of the effective address calculation as it is for all instructions.

Timing

PDP-6 processors are not synchronized by a clock. Therefore, PDP-6 systems run asynchronously with as high a duty factor as possible for each unit. Instruction times depend on several variables, such as the speed of the memory module in which the desired information is stored, whether the module is busy when the information is requested, and where the results are to be stored. Speed of execution also depends on the number of iterations involved in the effective address calculation, particularly since a new memory reference is required each time an indirect hit is encountered.

Table 2 summarizes instruction times for each group of instructions. Two times are given, a fast time and a slow time. The fast times are based on starting with instruction and data in fast memory. The slow times are based on starting with both instruction and data in the same core memory and allow for one index reference. The fast times are not necessarily minimum, since instructions in the immediate mode may run faster. Nor are the slow times maximum times, since an instruction may take considerably longer if there are several levels of indirect addressing. Exact times depend on the program context in which the instructions occur and on other factors. Therefore the figures should not be used to calculate program running time.

TABLE 2 SUMMARY OF INSTRUCTION TIMES

Instructions	Fast	Slow
Full and half word moves	1.9	4.0
Full and half word immediate	1.5	2.7
Byte manipulation	5.7	8.0
Byte manipulation and increment	7.2	8.0
Block transfer	$1.5+0.8n$	$2.4+1.2n$
Exchange	2.8	4.0
Fixed point add	2.7	4.3
Fixed point subtract	2.9	4.5
Fixed point multiply	14.5	16.1
Fixed point divide	23.4	25.0
Floating point add	5.8	8.0
Floating point subtract	6.0	8.2
Floating point multiply	12.4	14.5
Floating point divide	18.4	20.5
Boolean	2.7	4.3
Shifting (18 bits)	4.7	5.9
Memory, AC modification and testing	2.6	3.9
Arithmetic compare	2.7	4.4
Logical compare	2.7	4.4
Jumping	1.8	3.0
I/O		
basic	3.0	6.2
augmented	3.8	7.0
Push down	3.1	6.4

The flow diagram, Figure 8, can be used to calculate the approximate execution time of any instruction under any set of conditions.
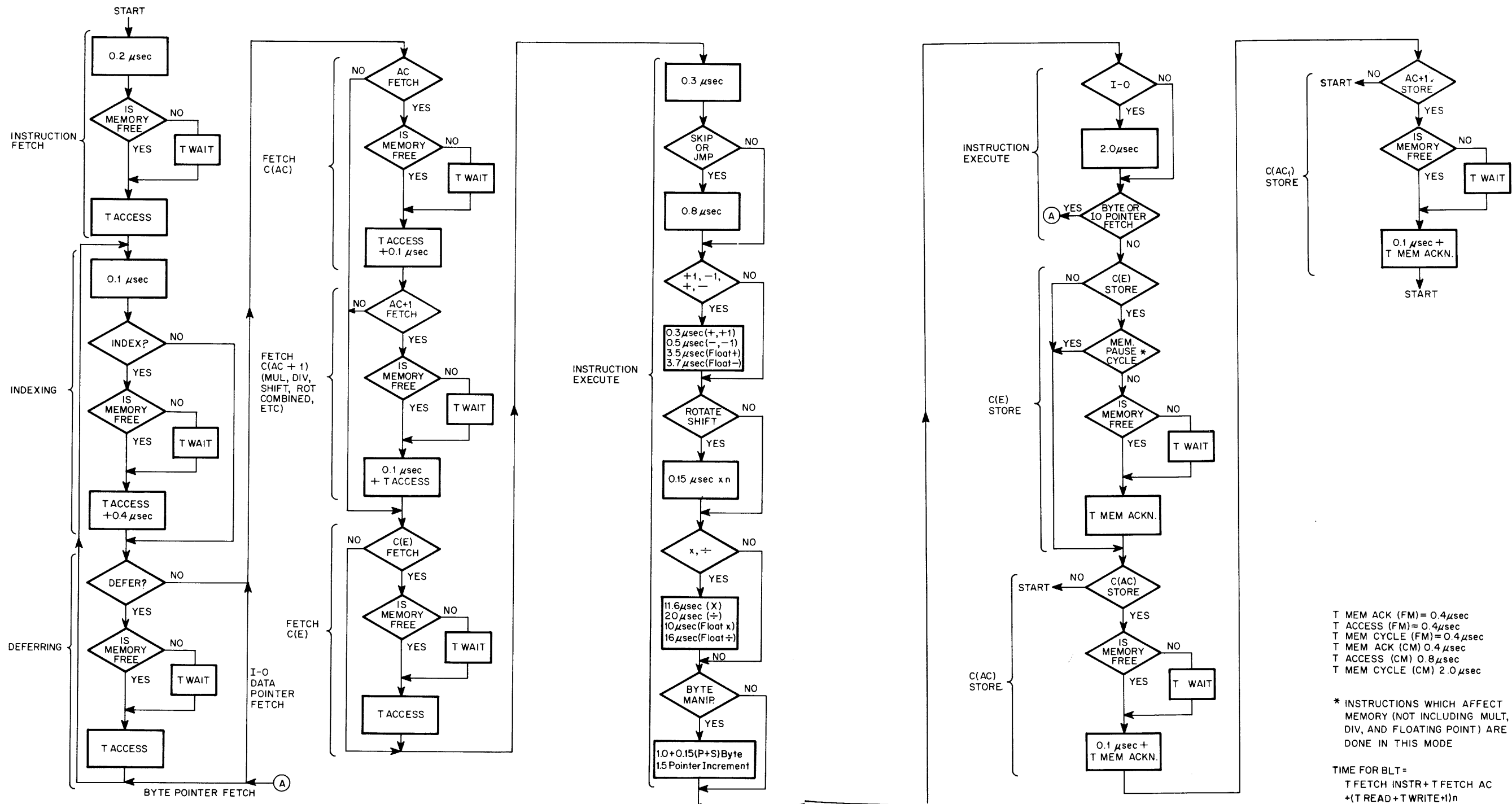
Figure 8   Instruction   Time Flow Diagram

INSTRUCTION CLASSES

Data Transmission

For all transmission instructions, the word whose contents are being moved is referred to as the source word. The word to which the data is being moved is referred to as the destination word.

FULL WORD TRANSMISSION

MOVE Move full word.

MOVS Move full word with left and right halves exchanged.

MOVN Move negative (twos complement) of full word.

MOVM Move magnitude of full word.

MODES

The source words are unaffected except in the S mode where the source word and the destination word are identical.

 Move contents of effective address to accumulator.

I Move effective address to accumulator.

M Move contents of accumulator to effective address.

S Move contents of effective address, swapped, negated, or set to magnitude, to effective address.

HALF WORD TRANSMISSION

There are 16 half word move operations derived from all possible combinations of left to left, right to right, left to right, and right to left, with four possible operations on the other half of the destination word. The operations are: do nothing to the other half; set the other half to all zeros; set the other half to all ones; and set all bits of the other half equal to the high order bit of the half being moved.

HLL Move left half to left half, no effect on right half of destination word.

HRR Move right half to right half, no effect on left half of destination word.

HRL Move right half to left half, no effect on right half of destination word.

HLR Move left half to right half, no effect on left half of destination word.

HLLZ Move left half to left half, set right half of destination word to all zeros.

HRRZ Move right half to right half, set left half of destination word to all zeros.

HRLZ Move right half to left half, set right half of destination word to all zeros.

HLRZ Move left half to right half, set left half of destination word to all zeros.

HLLO	Move left half to left half, set right half of destination word to all ones.
HRRO	Move right half to right half, set left half of destination word to all ones.
HRLO	Move right half to left half, set right half of destination word to all ones.
HLRO	Move left half to right half, set left half of destination word to all ones.
HLLE	Move left half to left half, set all bits of the right half to the sign (bit 0) of the left half.
HRRE	Move right half to right half, set all bits of the left half to the sign (bit 18) of the right half.
HRLE	Move right half to left half, set all bits of the right half to the sign (bit 0) of the left half.
HLRE	Move left half to right half, set all bits of the left half to the sign (bit 18) of the right half.

MODES

The source words are unaffected except in the S mode where the source word and the destination word are identical.

	Move contents of effective address to accumulator.
I	Move effective address to accumulator. (If source is left half, a 0 is transferred.)
M	Move contents of accumulator to effective address.
S	Move contents of effective address to effective address.

BYTE MANIPULATION

Byte manipulation instructions permit easy access to any number of contiguous bits located anywhere in a single, 36-bit memory word. To specify the size and location of the byte, a pointer word is located by the effective address of the byte manipulation instructions. The I, X, and Y fields of the pointer word (see Figure 9) are used in the usual manner to compute an effective address which will be the location of the memory word containing the byte. The P field specifies the number of bits between the right end of the word and the farthest right bit of the byte. The S field specifies the size of the byte, up to 36 bits. Therefore, the byte is located in bits 36-P-S through 35-P.
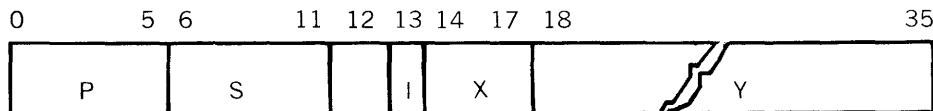
Figure 9 — Byte Pointer Format

To store successive bytes in memory and retrieve them with a minimum of instructions, the processor must be able to increment the pointer word. To increment, the size of the byte is subtracted from P, moving the position to the right by one byte. If there is insufficient room in the memory word for the next byte (P-S < 0), then the Y field of the pointer word is incremented by one and P is reset to 36-S.

LDB	Load bits 35 to 36-S of the accumulator with the specified byte. Bits 0 through 35-S of the accumulator are cleared.
DPB	Deposit byte, starting in bit 36-S through bit 35 of the accumulator, in the portion of the memory word as specified by the pointer word.
IBP	Increment byte pointer by one byte position.
ILDB	Same as LDB, but increments pointer word before loading.
IDPB	Same as DPB, but increments pointer word before depositing.
	See the BIS flag for its effect on the increment instruction.

MISCELLANEOUS

EXCH	The contents of the effective address and the contents of the specified accumulator are exchanged.
BLT	Moves a block of storage (source block), beginning at the location specified by the left half of the designated accumulator, to a block (destination block) beginning at the location specified by the right half of the accumulator. The size of the block is determined by the effective address; that is, the effective address is equal to the last location of the destination block. The source block is unaffected unless it overlaps the destination block. The accumulator is only referenced at the beginning of the instruction; therefore, it is not changed unless a program interrupt occurs. If a program interrupt occurs, the left half of the accumulator (source block) is set to the address of the next word to be moved, and the right half is updated to the next word in the destination block. When the break is dismissed, the BLT instruction then resumes where it left off.

Arithmetic and Logical

Two types of arithmetic are used by the Type 166 Processor: fixed point and floating point. Integer arithmetic, such as address and index arithmetic, is only a special case of fixed point arithmetic. For consistency and to make possible the use of fixed point arithmetic compare commands with floating point number, all arithmetic in the 166 is done in twos complement binary. That is, carries from the highest order bit are dropped, and negative numbers are the twos complement of the positive numbers with the same magnitude.

To form the negative (twos complement) of a number, change all zeros to ones and all ones to zeros, then add one to the result. In this scheme, there are two points to be carefully noted. Bit zero will always function as a sign bit; if zero, the number is positive, if one, the number is negative. (However, this must not lead to identification with sign-magnitude conventions.) Secondly, forming the negative by twos complement applies to floating point numbers as well as fixed point numbers. Therefore, the exponent part of floating point numbers, as well as the fractional part, is complemented.

Fixed point numbers are represented in binary form so that the value of a number in memory is:

$$\sum b_i 2^{(35+n-i)}$$

where b_i designates the contents of bit i, and n depends on the position of the binary point. However, since the position of the binary point is of no consequence to the machine, two vastly different numbers may have the same representation in memory. It is left to the programmer to establish consistent point conventions for any sequence of operations

Floating point numbers use 1 bit for the sign, 8 bits for the exponent, and 27 bits for the fraction. The binary point is therefore to the left of bit 9.
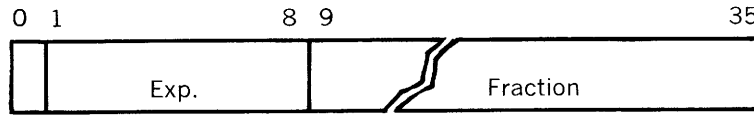
Figure 10— Floating Point Format

For positive numbers, the exponent is excess 200_8. Since in Type 166 negative numbers are always the twos complement of the positive number with the same magnitude, exponents for negative numbers are the ones complement of the excess 200_8 exponent. Floating point zeros are represented by all zeros. Floating point numbers are considered to be normalized if the sign bit does not equal bit 9, or if bits 9–35 contain 400000000_8.

FIXED POINT ARITHMETIC

Fixed point arithmetic is single precision. However, to facilitate programming double precision, a carry from bits 0 and 1 will set the CRY0 and CRY1 flags, respectively. Also, the fixed point instructions will set the OV flag if the sign of the answer is not correct with respect to the signs of the operands. For additional conditions under which the OV flag will be set, see the IMUL, DIV, and IDIV instructions. Note that the CRY0 and OV flags are not set under equivalent conditions.

ADD   The memory operand is added to the contents of the accumulator. This instruction may set the CRY0, CRY1, and OV flags.

SUB   The memory operand is subtracted from the contents of the accumulator. The subtraction is carried out by adding the twos complement of the contents of the accumulator to the memory operand, followed by forming the twos complement of the result. This instruction may set the CRY0 ,CRY1, and OV flags.

MUL   Multiply the contents of the accumulator by the memory operand. The contents of the designated AC are placed in the MB. The contents of the effective address are placed in the MQ. The AR is cleared.

For multiplication the MQ has an extra low order bit, bit 36, which is cleared before multiplication begins. Bits shifted out of MQ_{35} enter MQ_{36}.

The multiplication algorithm is:

1. If MQ_{35} and MQ_{36} are equal, then shift AR/MQ right one place.
2. If MQ_{35} is a 1, and MQ_{36} is 0, then subtract C(MB) from the AR. Shift AR/MQ right one place.
3. If MQ_{35} is a 0 and MQ_{36} is a 1, then add C(MB) to the AR and shift AR/MQ right one place.

AR/MQ is treated as one 72-bit word. Bits shifted out of AR_{35} enter MQ_0. The algorithm is repeated until the bit originally in MQ_0 reaches MQ_{35}. The algorithm is then repeated once more with shift MQ right one place replacing shift AR/MQ. That is, the final shift is for the MQ only with AR_0 being copied into MQ_0 so the signs of the AR and the MQ agree.

IMUL   Is the same as MUL, except that the low order part of the product is stored as specified by the mode control. The high order part is lost. This instruction will set the OV flag on the further condition that there are significant bits in the high order part of the product.

DIV   Divide the dividend by the contents of the memory operand. The dividend is a 72-bit word with the high order part set equal to the contents of the designated accumulator n, and the low order part set equal to the contents of accumulator n + 1. If one or both of the operands is negative, the effect of this instruction is as if division were performed on the magnitudes of the operands. The answer is then formed by twos complementing where necessary. The remainder has the sign of the dividend, and the sign of the quotient is the exclusive OR of the sign of the dividend and the sign of the divisor. The quotient is stored as specified by the mode control. The remainder, except in the M mode, goes to accumulator n + 1. The instruction sets the OV flag if the high order part of the dividend is greater than or equal to the divisor.

IDIV   Same as DIV, except that the low order part of the dividend is set to the contents of the designated accumulator. The high order part of the dividend is set to the sign bit of the low order part. OV is possible only if the divisor is zero.

MODES

Combine contents of the effective address with the contents of the accumulator; results to accumulator.

I   Combine the effective address with the contents of the accumulator; results to accumulator.

M   Combine contents of effective address with the contents of the accumulator; results to the effective address.

B   Combine contents of the accumulator with the contents of the effective address; results to both accumulator and effective address. If the effective address equals the accumulator number plus one, the effective address is left with either the low order part of the result or with the remainder, since the accumulators are stored after the memory.

FLOATING POINT ARITHMETIC

The floating point instructions may set the OV flag if exponent overflow or underflow occurs. No attempt is made to distinguish between them.

CAUTION

The floating point instructions may not give the
correct results for unnormalized operands.

FAD   Floating add the memory operand to the contents of the accumulator. The number with the smaller exponent is shifted right by the difference in the magnitude of the exponents. The fractional parts are then added as fixed point numbers. The result is normalized, and the answer given in terms of the higher exponent. This instruction may set the OV flag.

FSB   Floating subtract the memory operand from the contents of the accumulator. The subtraction is done by adding, as above, the twos complement of the contents of the effective address. May set OV flag.

FMP   Floating multiply the contents of the accumulator by the memory operand. The multiplication is done by adding the exponents and multiplying the fractional parts as fixed point numbers. May set the OV flag.

FDV   Floating divide the contents of the accumulator by the memory operand. The division is done by subtracting the exponent parts and dividing the fractional parts as fixed point numbers. May set the OV flag.

There are four instructions which include rounding. The rounding is done as follows: If bit 1 in the low order part of the result is a 1, and the sign of the high order part is positive, then add one to the lowest order bit in the high order part. Otherwise, do nothing.

FADR   Same as floating add, followed by rounding.

FSBR   Same as floating subtract, followed by rounding.

FMPR   Same as floating multiply, followed by rounding.

FDVR   Same as floating divide, followed by rounding.

MODES

Combine contents of effective address with contents of accumulator; results to accumulator.

L   Combine contents of effective address with contents of accumulator; results to accumulator, remainder or low order bits to AC + 1, left adjusted with no exponent part.

M   Combine contents of effective address with contents of accumulator; results to memory.

B   Combine contents of effective address with contents of accumulator; results to accumulator and to effective address.

There is one floating point instruction which has no mode control:

FSC   (Floating Scale) Multiply contents of accumulator by 2^E, where E is the effective address.

BOOLEAN

The Type 166 Processor can perform all 16 Boolean operations of two variables. The contents of the designated accumulator and the memory operand are matched on a bit for bit basis. The result is a 0 or 1 in the corresponding bit of the resultant word. The Boolean instructions are listed below. The result of each instruction is shown at the right under the four possible memory-accumulator configurations.

Mnemonic	Meaning	Memory Bit Accumulator Bit	0 0 1 1 0 1 0 1
SETZ	Set to all zeros		0 0 0 0
AND	And		0 0 0 1
ANDCA	AND with the contents of the accumulator complemented		0 0 1 0
SETM	Set the result equal to the contents of the effective address		0 0 1 1
ANDCM	AND with the contents of the effective address complemented		0 1 0 0
SETA	Set the result equal to the contents of the accumulator		0 1 0 1
OR	Exclusive OR		0 1 1 0
IOR	Inclusive OR		0 1 1 1
ANDCB	AND with the contents of both the accumulator and the effective address complemented		1 0 0 0
EQV	Equivalent		1 0 0 1
SETCA	Set the resultant to the complement of the contents of the accumulator		1 0 1 0
ORCA	Inclusive OR with the contents of the accumulator complemented		1 0 1 1
SETCM	Set the resultant equal to the complement of the contents of the effective address		1 1 0 0
ORCM	Inclusive OR with the contents of the effective address complemented		1 1 0 1
ORCB	Inclusive OR with the contents of both the accumulator and the effective address complemented		1 1 1 0
SETO	Set resultant to all ones		1 1 1 1

MODES

	Combine contents of effective address with contents of accumulator; results to accumulator.
I	Combine effective address with contents of accumulator; results to accumulator.
M	Combine contents of effective address with contents of accumulator; results to effective address.
B	Combine contents of effective address with contents of accumulator; results to both accumulator and effective address.

SHIFTING

There are three kinds of shifting:

1) arithmetic shift — performs twos complement multiplication by powers of 2. The sign is unchanged. When going to the right, the sign is shifted into bit 1. Ones or zeros leaving bit 35 are lost. The Overflow flag is set if, during shifting, the exclusive OR of $C(AR)_0$ and $C(AR)_1$ becomes a 1.

2) rotate — bits leaving one end enter at the other end.

3) logical shift — bits leaving one end are lost, and zeros enter the other end.

The direction and number of places to be shifted are determined by bits 18 and 28-35 of the effective address, which together form a 9-bit twos complement number. If this number is positive, the shift is to the left; if negative, the shift is to the right. The number of bits shifted is equal to the magnitude of the number.

Shifts may be of the contents of an accumulator A or of a combined word, consisting of the contents of the accumulator A and the contents of the accumulator A + 1, in that order. If a combined shift instruction, and the specified accumulator is 17_8, then location 20_8 will be rotated unless locations 20_8-37_8 are read-only memory. Consequently, it is best to avoid this combination.

ASH	Arithmetic shift
ROT	Rotate
LSH	Logical shift
ASHC	Same as ASH, except that the contents of accumulator n and accumulator n + 1 are treated as one 72-bit word with accumulator n on the left or higher order position. When shifting either left or right, the sign bit of accumulator n + 1 is set equal to the sign of accumulator n. When shifting right, bits moving through bit 35 of accumulator n enter bit 1 of accumulator n + 1. When shifting left, ones or zeros leaving bit 1 of accumulator n + 1 enter bit 35 of accumulator n. Bits leaving bit 1 of accumulator n are lost. Zeros enter bit 35 of accumulator n + 1 from the right.
ROTC	Rotate combined accumulator
LSHC	Logical shift of combined accumulator

Executive

MEMORY AND ACCUMULATOR MODIFICATION AND TESTING

There are two sets of instructions from this group: accumulator jumps and memory skips. The jump instructions test and/or modify contents of the accumulator. If the conditions specified by the mode control are met, program control jumps to the effective address. Skip instructions test and/or modify the contents of the effective address. If the conditions specified by the mode control are met, the next instruction is skipped.

Memory Skips	Accumulator Jumps	
SKIP	JUMP	No modification, test only
AOS	AOJ	Add one and test
SOS	SOJ	Subtract one and test

MODES

Skip or jump if the contents of the word tested are:

—	Never skip
L	Less than zero
E	Equal to zero
LE	Less than or equal to zero
A	Always skip
GE	Greater than or equal to zero
N	Not equal to zero
G	Greater than zero

ARITHMETIC COMPARE

CAM   Compares algebraically the contents of the effective address with the contents of the accumulator. If the condition specified by the mode is met, the next instruction is skipped.

CAI   Same as CAM, except that it compares the effective address with the contents of the accumulator.

MODES

—	Never skip
L	Skip if contents of accumulator are less than contents of data word.
E	Skip if contents of accumulator are equal to contents of data word.
LE	Skip if contents of accumulator are less than or equal to contents of data word.
A	Always skip.
GE	Skip if contents of accumulator are greater than or equal to contents of data word.
N	Skip if contents of accumulator are not equal to contents of data word.
G	Skip if contents of accumulator are greater than contents of data word.

LOGICAL COMPARE AND MODIFY

Bits of an accumulator word that are masked by bits of a memory word, that is, those bit positions of the accumulator corresponding to bit positions in the memory word containing ones, can be modified and/or tested to determine a skip. The masked bits may be left unchanged, set to zeros, set to ones, or complemented. The masking may be done by the contents of the memory word, specified by the effective address taken directly; by the contents of the memory word with its left and right halves swapped; or by the effective address itself. Since the effective address is only 18 bits, it must be specified whether it is to mask the left or right half of the accumulator. The unused half of the accumulator is masked by zeros.

The conditions under which a skip will occur are specified by the mode control. The test is made before any modification occurs. Instructions set the PC Change flag if a skip is executed.

TDN	Test, masking by direct contents of memory word, no modification.
TSN	Test, masking by swapped contents of memory word, no modification.
TLN	Test, masking left half by effective address, no modification.
TRN	Test, masking right half by effective address, no modification.
TDZ	Test, masking by direct contents of memory word, set masked bits to zero.
TSZ	Test, masking by swapped contents of memory word, set masked bits to zero.
TLZ	Test, masking left half by effective address, set masked bits to zero.
TRZ	Test, masking right half by effective address, set masked bits to zero.
TDO	Test, masking by direct contents of memory word, set masked bits to ones.
TSO	Test, masking by swapped contents of memory word, set masked bits to ones.
TLO	Test, masking left half by effective address, set masked bits to ones.
TRO	Test, masking right half by effective address, set masked bits to ones.
TDC	Test, masking by direct contents of memory word, complement masked bits.
TSC	Test, masking by swapped contents of memory word, complement masked bits.
TLC	Test, masking left half by effective address, complement masked bits.
TRC	Test, masking right half by effective address, complement masked bits.

MODES

Skip if:

—	Never skip
E	All masked bits are zero
A	Always skip
N	Not all masked bits are equal to zero

JUMP INSTRUCTIONS

JSR   Jump to subroutine. The AR OV, AR CRY0, AR CRY1, PC Change, BIS, and User Mode flags are all stored in the contents of the effective address, bits 0 through 5 respectively, and the BIS flag is cleared. The program counter, which has been incremented to point to the next instruction in sequence, is stored in C(E), bits 18–35. Program control then jumps to the effective address plus one. (See JRST for the return instruction.)

JSP   Jump and save program counter. The contents of the program counter, which will point to the next instruction, are stored in the designated accumulator. Program control jumps to the effective address. (The return from this jump can be a single jump instruction using the accumulator as an index register.)

JSA   Jump and save accumulator. The accumulator is stored in the effective address. The PC, which points to the next instruction is stored in the right half of the AC, and the effective address is stored in the left half of the accumulator. Program control jumps to the effective address plus one.

JRA   Jump and restore accumulator. This is the return instruction to match JSA. The contents of the accumulator are replaced by the word addressed in its left half. Control is transferred to the effective address.

There are three reasons for the JSA-JRA pair: to provide for subroutines with multiple entries; to provide an easily accessible reference for getting data; and to prevent loss of information, making it possible to nest subroutines.

AOBJP   Increment both halves of accumulator by one, and jump if the result is positive. (See the Miscellaneous Instructions.)

AOBJN   Increment both halves of accumulator by one, and jump if the result is negative. (See the Miscellaneous Instructions.)

JRST   Jump and restore. (See the Miscellaneous Instructions.) Bits 9–12 (accumulator field) determine the effect of this instruction as follows:

If bit 9 = 1, (AC8), reset the current priority interrupt channel.

If bit 10 = 1, (AC4), halt.

If bit 11 = 1, (AC2), restore flags (see note below).

If bit 12 = 1, (AC1), set User Mode flag to a 1 (enter user mode).

This instruction, with bit 11 equal to 1, may be used as a return instruction to match the JSR. In this case the indirect bit, bit 13, must also be set to 1. Otherwise, the instruction is undefined.

The AR OV, AR CRY0, AR CRY1, PC Change, and BIS flags are set to the C(E), bits 0–4 respectively. If C(E), bit 5, is a 1, the User Mode flag is set; otherwise, the User Mode flag is unchanged.

JFCL   If designated flags are set, jump and clear flags. (See the Miscellaneous Instructions.) The flags are designated as follows:

Bit 9, (AC8), designates the OV flag.

Bit 10, (AC4), designates the CRY0 flag.

Bit 11, (AC2), designates the CRY1 flag.

Bit 12, (AC1), designates the PC Change flag. This flag remains cleared.

The PC Change flag is set whenever the PC is incremented by more than one to enable program flow tracing.

MISCELLANEOUS

XCT   Execute instruction at effective address. The instruction at the effective address may be another XCT instruction. Thus, chains of XCT instructions are permitted. (See the Miscellaneous Instructions.)

Push Down Instructions

In the following group of instructions, note that if a carry from bit 0 occurs when incrementing (decrementing) the accumulator, the PDL OV flag is set.

PUSH   The contents of the effective address are moved to the next location on the push down list, that is, the location to which the address part of the accumulator is pointing + 1. The accumulator is then incremented by 1000001_8.

PUSHJ   The accumulator is incremented by 1000001_8. The contents of the program counter, which is pointing to the next instruction, are then moved to the next free location on the push down list, that is, the location to which the address part of the accumulator is pointing. Control then jumps to the effective address. This instruction sets the PC Change flag.

POP   The last word placed on the push down list, that is, the contents on the location to which the address part of the accumulator is pointing, is moved to the effective address. The accumulator is then decremented by 1000001_8.

POPJ   The last word placed on push down list, that is, the contents of the location to which the address part of the accumulator is pointing, is moved to the program counter. The accumulator is then decremented by 1000001_8. This instruction sets the PC Change flag.

Input-Output Instructions

CONO   The effective address is transferred to the device control register. For the significance of the bit configuration see the specific device write-up .

CONI   The contents of the device control register are transferred to the contents of the effective address.

DATAO   The contents of the effective address are placed on the I/O bus for output by the device. This instruction will also cause certain control actions in the device; see the particular device for details.

DATAI   Data read in by the device and available in the device buffer register is transferred to the effective address. This instruction will also cause certain control actions in the device; see the particular device for details.

CONSZ   This instruction is equivalent to a CONI followed by a test and skip. If the logical AND of the contents of the device control register and the contents of the effective address equals zero, then skip. Otherwise, execute the instruction following the CONSZ.

CONSO   Equivalent to a CONI followed by a test and skip. If the logical AND of the contents of the device control register and the contents of effective address is not equal to zero, then skip. Otherwise, execute the instruction following the CONSO.

BLKI   This instruction is used to read in a block of data. It uses the effective address to locate a pointer word, then increments both halves of the pointer word by one. The right half of the pointer word is used to determine the address of the memory location which is to receive the input data.

The instruction reads in the data word from the appropriate I/O device in exactly the same way as a DATAI instruction. The left half of the pointer word is then tested for a zero (end of block condition). If zero, control goes to the next instruction in sequence. If not zero, then

      a)   if in priority interrupt, dismiss channel.

      b)   if not in priority interrupt, skip the next instruction.

BLKO   This instruction is exactly like the BLKI, except that it is an output instruction.

INPUT-OUTPUT PROGRAMMING FOR TYPE 166 PROCESSOR

This section is concerned with the use of the Type 166 Processor as an I/O processor. For details of the I/O system see Chapter 4. Here we will be concerned only with the use of the 166 I/O commands along with the use and control of the 166 Priority Interrupt System.

There are four basic 166 instructions for controlling the I/O devices; that is, for generating the six command pulses described in Chapter 4. They are:

      CONO   Generates the conditions out sequence.

      CONI   Generates the conditions in sequence.

      DATAO   Generates the data out sequence.

      DATAI   Generates the data in sequence.

Of the remaining four commands two are equivalent to CONI followed by an instruction to the processor to test and skip. They are CONSZ and CONSO. The remaining two, BLKO and BLKI, are instructions to the processor to control the repeated execution of DATAO and DATAI commands to the I/O system.

The use of these instructions will be illustrated by programming examples for a generalized device. For their use with actual devices, a device number must appear in bits 3–9 of the instruction word. (See the I/O Instruction Format, Figure 7.) This number may be generated by a device mnemonic.

Further, the significance of the bits in the effective address used with a CONO instruction and the significance of the bits read into the contents of the effective address by a CONI instruction must be known for each device. Finally, due consideration must be given to the number of bits which can be transmitted to the data register since information to be transmitted from memory to the device must appear in the n low order bits of the memory word, where n is the length of the data register. Similarly, due consideration must be given to the number of bits which the device will transmit to memory through the processor. All of this required information may be found by consulting the individual device write-up in Chapter 4.

Up to 126 I/O devices may be attached to the 166 through an I/O bus. The reason for the restriction to 126 devices, rather than 128, is that the processor and the processor's priority interrupt system are considered to be two I/O devices as follows:

Processor

Device Number   000 000 0
Mnemonic   APR

The following I/O instructions may be used to refer to the processor:

DATAO   Used to set the memory protection and relocation registers. The contents of the effective address, bits 18–25, go to the relocation register; bits 27–34 go to the protection register.

DATAI   Reads the 36 data switches.

CONO   Bits 18–35 of the effective address have been assigned the following significance:

     18   If a 1, reset the PDL OV flag.

     19   I/O reset.

     22   If a 1, reset the Memory Protection flag.

     23   If a 1, reset the nonexistent Memory flag

     24   If a 1, turn the Clock Count Enable flag off.

     25   If a 1, turn the Clock Count Enable flag on.

     26   If a 1, turn the Clock flag off.

     27   If a 1, turn the PC Change Enable flag off.

     28   If a 1, turn the PC Change Enable flag on.

29	If a 1, turn the PC Change flag off.
30	If a 1, turn the OV flag enable off.
31	If a 1, turn the OV flag enable on.
32	If a 1, turn the OV flag off.
33–35	Assign a priority channel to the above processor flags.

If a priority channel is assigned to the processor flags, an interrupt will occur on that channel if any one, or more, of the following conditions hold:

1. If the Illegal Instruction flag is a 1.

2. If the non-existent Memory flag is a 1.

3. If the Clock Count Enable flag is a 1, and the Clock Count flag is a 1.

4. If the PC Change flag enable is a 1, and the PC Change flag is a 1.

5. If the OV flag enable is a 1 and the OV flag is a 1.

The real time clock runs at the power line frequency. It sets the Clock flag every 1/60 of a second for 60-cycle current whether or not the Clock Count Enable flag is on.

CONI   Bits 18–35 of the location specified by the effective address are set as follows:

18	To a 1 if the PDL OV flag is set.
22	To a 1 if the Illegal Instruction flag is set.
23	To a 1 if the non-existent Memory flag is set.
25	To a 1 if the Clock Count enable is on.
26	To a 1 if the Clock Count flag is set.
28	To a 1 if the PC Change enable is on.
29	To a 1 if the PC Change flag is set.
31	To a 1 if the OV enable is on.
32	To a 1 if the OV flag is set.
33–35	Set to the priority channel assignment.

Priority Interrupt System

Device Number 004_8 (000 000 1)

Mnemonic   PRS

The 166 Processor will honor interrupt requests on a priority basis at any time between the beginning of an instruction cycle and the end of the effective address calculation. However, if the instruction being executed is the result of an interrupt and is at one of the locations 40 + 2J, it can not be interrupted at any time during the instruction cycle. The processor honors the request by executing the instruction in location 40 + 2J, where J is the channel number. The program counter is not affected by the interrupt unless the trapped instruction is a jump instruction.

If a jump, it is left to the programmer to save the program counter if he has any intention of returning to the interrupted program sequence.

It is intended that the instructions executed by an interrupt, that is, the instructions located in locations 40 + 2J, should be either BLKI, BLKO, JSR, or XCT which executes one of the preceding three. While other instructions are not illegal, their results could be unfortunate. For example, the contents of the program counter could be lost, making it impossible to return to the main program.

The priority interrupt system is controlled by the CONO and CONI instructions by referring to device number 000 000 1 and setting bits in the effective address as follows:

CONO

23	If 1, clear the priority interrupt system.
24	Activate an interrupt on the channels selected by bits 29–35.
25	If 1, turn on the channels selected by bits 29–35.
26	If 1, turn off the channels selected by bits 29–35.
27	If 1, turn off the priority interrupt system.
28	If 1, turn on the priority interrupt system.
29–35	If a bit is 1, it selects the corresponding channel, with channel 1 in bit 29 to channel 7 in bit 35.

CONI

| 28 | If 1, the priority interrupt system is on. |
| 29–35 | If a bit is 1, the corresponding channel is on. |

Once an interrupt has occurred on a given channel, the channel must be dismissed by a BLKI or BLKO instruction, or a JRST or CONO instruction. (See the instruction descriptions for details.) If the break is not dismissed, there will be no further interrupts on lower priority channels. While servicing a device, the channel may be interrupted by a request on a higher priority channel. If so, the service to the lower priority device can be resumed by the instructions which dismissed the channel.

Input-Output Programming

Since the purpose of this section is merely to discuss the use of the I/O operations of the Type 166 Processor, particular I/O devices are not considered. Rather, the discussion and programming examples are presented for a generalized device. All that is required is that the device be defined sufficiently like the devices found in a PDP-6 system.

For some degree of definiteness, consider a generalized input device. The device is like those for the PDP-6 system in that it has a control register and a data register. The operation of the device is completely specified by bits of the control register which may be set by a CONO instruction. Further, the device control module can set some of the bits to reflect the status of the device, and the device may be requested to place the contents of the control register on the data lines by a CONI instruction. Finally, a DATAI command will cause the device to place data from its data register on the data lines for transmission to the processor.

The device may be turned on to read one character by setting the Busy flag (a designated bit in the control register) to 1. When the character is ready in the data register, the device clears the Busy flag to 0 and sets the Done flag (also a designated bit in the control register) to 1. A DATAI instruction will result in the contents of the data register being transferred through the data lines to the processor followed by resetting the Busy flag to read the next character and clearing the Done flag.

If a priority channel has been assigned by a CONO instruction, which places a channel number in the low order three bits of the control register, setting the Done flag to 1 will cause an interrupt on the assigned channel.

When using I/O instructions, we must include a device number or a device mnemonic. In the examples which follow for the generalized device, we will use the mnemonic MNE.

Such a device could be programmed without the priority interrupt system as follows:

    CONO, MNE, Y

    CONSO, MNE, X

    JRST, .—1   .—1 means this location minus one.

    DATAI, MNE, E

The CONO instruction transmits the effective address Y to the device control register to set the Busy flag and to specify the operation. CONSO matches the effective address X against the contents of the control register to detect when the Done flag is set to 1. JRST is an unconditional jump to CONSO. When the Done flag is set, the program skips to the DATAI instruction which reads the data in and moves it to location E in memory.

In the priority interrupt mode, the same operation can be carried out as follows. The priority interrupt system and channels are turned on by a CONO instruction. Then, using channel 1 we could write the following program:

    CONO, MNE, Y + 1, + 1 to assign PI channel at location 42,
    and beginning at INPUT

    INPUT, EMPTY REGISTER FOR PROGRAM COUNTER AND FLAGS

    DATAI, MNE, E

    JRST, 12 @ INPUT

Where @ is an assembler symbol meaning jump indirectly to the location specified by the right half of location INPUT; that is, resume the interrupted instruction.

In this way the program could continue without waiting for the device. As soon as possible after setting the Done flag, the JSR at location 42 is executed. The control jumps to INPUT +1 with the flags and C(PC), still pointing to the location of the interrupted instruction, stored in INPUT. The DATAI is executed followed by the JRST. The number 10 in the accumulator field sets bits 9 and 11 to dismiss the priority channel and to restore the flags and program counter. It is important to note that, unless bit 9 is set, the channel will not be dismissed and it will be impossible for an interrupt to occur on all lower priority channels. Further, it may be desirable to add an additional CONO instruction to disconnect or disable the priority channel. Otherwise, additional interrupts will occur each time the device finishes reading in a new character.

To read in a block of information, the BLKI instruction may be used. Assuming the pointer for the BLKI has been properly set and is at location E, that is, the left half

of location E contains the negative of the size of the block and the right half contains the location of the first word of the block minus one, there are two options. Without the priority interrupt:

CONO, MNE, Y

CONSO, MNE, X

JRST, .—1   .—1 means this location minus one.

BLKI, MNE,   E

JRST, MNE, .+2   .+2 means this location plus two.

JRST, .—4   .—4 means this location minus four.

When the entire block has been read in, BLKI causes the next instruction to be executed. Until then, it skips.

Using the priority interrupt system:

CONO, MNE, Y + 1, + 1 to assign channels at locations
42 and 43:

42, BLKI, MNE, E

JSR, MNE, OFF

The BLKI instruction will automatically dismiss the priority channel after the word of data has been read in unless it is the last word of the block. If the last word, the JSR following the BLKI is executed. The sequence of instructions to which the JSR jumps must then dismiss or turn off the interrupt.

At location OFF a blank register could be followed by a sequence of one or more CONO instructions to turn off the device and/or priority channel and/or a JRST instruction.

Several devices can be assigned to the same channel, such as several Teletype stations in a time-sharing system. For this situation the instruction, in location $40 + 2j$, executed by the interrupt is a JSR to a subroutine. The subroutine consists of a sequence of condition skip instructions which examine the control register of each device assigned to the channel and jump to instructions to service the device whose Done flag is set. If only one device is serviced, and more than one is calling for an interrupt, then the other devices will immediately call for an interrupt when the channel is dismissed.

So far I/O programming has been discussed for a generalized input device. Programming for output is sufficiently similar to be given no separate detailed explanation. However, there are some minor differences. For example, rather than turning on the device with the same CONO instruction that sets the mode of operation, priority channel, etc. and waiting for the Done flag to give a data transfer command, with an output device we set the mode of operation with CONO and let the first DATAO instruction turn on the device and set the Busy flag. Further, the meaning of an interrupt is different for an input and output device. To be specific, consider the transfer of a block of data using the BLKI and BLKO instructions. When the JSR instruction following a BLKI is executed, the device has finished its present assignment. If, for some reason, it is desired that it not get the next word of data, a CONO instruction may be given immediately to clear the Busy flag. In contrast, when the JSR following a BLKO instruction is executed, the DATAO command has just been given, and the device will probably still be busy. If some other action is desired, such as writing an end of file on tape, we must wait for the Done flag.

CHAPTER 3
MEMORY

PDP-6 memory system configurations can be of almost any type, depending on system need. One processor can address several memory modules, and up to four processors can address a single module. Each module can therefore be a component in up to four memory systems. The core memory controls have a priority system used to determine which processor gets service. One processor has the highest priority; a second one, the second highest; and the third and fourth processors alternately share the third priority. For the Type 162 Fast Memory, a manual switch selects one of the connected processors.

The basic characteristic of any configuration is that each processor is connected to its memory modules through a bus. A function of the module is to recognize its address, to read and write information, and to put appropriate control information and data on the bus for the processor. Once it has received its instructions, the module performs these functions independent of the processor. In reading or writing, once the data has been taken off of the bus, the bus is free to carry requests from the processor to other memory modules.

To a processor, the memory system appears as one homogeneous unit. The processor simply places the appropriate data and/or commands on its bus and awaits the response of the memory. Should another processor have a memory request in progress in the same module, the later request is delayed until the module is free. Up to 262,144 words of memory can be directly addressed by one processor.

PDP-6 memory modules are of two types: Type 163 Core Memory (8,192 or 16,384 words, 2-microsecond cycle time) and Type 162 Fast Memory (16 words, 0.4-microsecond cycle time). In all memory configurations connected to the Type 166 Processor bus, the first 16 locations in memory are used as accumulators, index registers, and ordinary memory locations. Since the processor makes constant reference to these 16 locations, the use of fast memory here causes a significant increase in system speed. Figure 11 shows a typical memory system.
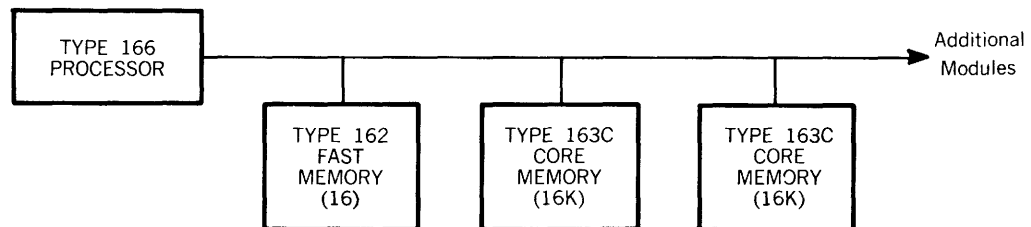
Figure 11   Typical Memory System

MEMORY FUNCTIONS

The processor can make three kinds of requests to memory: read, write, or read-pause-write. In each case the processor places the address of the module on the five module address lines and a request cycle signal on the request cycle line. It also specifies one of the functions to be performed, as described below. If the module is free, it returns an address acknowledge signal within 0.4 microseconds. If the module is busy, the processor waits until the module is free to acknowledge.

If the processor wants information from memory, it gives a read request, and when the data is available in the memory buffer, the memory module control places it on the bus. At the same time the memory module control sends a read restart signal to the processor, causing the processor to take the data off the bus. At this point the processor is free to continue its operations, which may be a new memory request, while the memory completes its cycle independently.

To write information into memory, the processor places the data on the bus along with the write request. As soon as the processor receives the acknowledge signal from the memory module, it is free to continue its operations. Again, this may include a new memory request.

The read-pause-write function can be used to advantage when the processor is to take data out of memory, perform a short operation (anything except multiply, divide, or floating point arithmetic), and store the results back in the same location. Read-pause-write is specified by requesting both read and write. When the processor receives the data and the read-restart signal, it performs the operation while the memory module remains connected. When the operation is completed, the processor places the results on the bus along with a write-restart command. It is then free to continue while the memory module completes its cycle.

ADDRESS SELECTION

Address selection consists of dividing the 18-bit address in the memory address register into a module address and a word address. The 5-bit module address is placed on the 5 module selection lines, and the word address is placed on the 14 address lines. As can be seen, the 5-bit module address and the 14-bit word address share 1 bit.

The exact method of determining the 5-bit module address depends on the system configuration. In general, four of the module address lines are connected to the four high order bits of the memory address register, while the fifth line is connected to any one of the fourteen low order bits. The fifth line is used in memory over-lapping, as explained below.

Each module may be set to respond to one or more designated addresses by jumper connections within the module interface. The addresses are determined at each interface separately. Therefore, during the course of multiple processor system opera-tion, different processors may address a given module by different sets of addresses.

A fast memory module added to the memory system replaces the first 16 locations in core memory for all processors connected to the fast memory module. When the fast memory module is addressed, bits 18 through 21, containing the module selection bits, are 0, and bits 22 through 31, the high order bits of the address selection, are

also 0. Sensing this condition, the processor sends a fast block signal to all memory modules, which enables the fast memory and inhibits the core memory. The first 16 locations in the first core memory module may, of course, be used by another processor not connected to a fast memory module.

MEMORY OVERLAPPING

In an asynchronous system it is highly desirable to interleave memory addresses in modules so that consecutive memory references are made to different modules.

For example, all even addresses can be assigned to one module and all odd addresses to another. The processor can then address a request to the other module without waiting for the previously addressed module to finish its cycle.

Since it is impossible, in general, to know where the next memory reference will be made in relation to the present one, there is no way to guarantee that the next reference will be to a new module. However, statistically there is a gain in performance if interleaved address ranges are assigned to different modules.

In PDP-6 memory systems the size of the interleaved blocks of memory locations can be any power of two from 1,024 to 8,192 depending on which of the 14 low order bits is connected to the fifth selection line. The module interfaces are set to respond to zero or one in the fifth position, depending upon the addressing scheme. When the fifth selection line is used in this manner, the module gets the word location part of the address from 13 of the 14 low order address bits and from 1 of the 4 remaining high-order selection bits.

MEMORY PROTECTION

A PDP-6 system can be concurrently shared by two or more users. In this case, the users' programs are assigned blocks of memory storage which may be any length from 512 to 262,144 locations in powers of two. Since an erroneous address in one user's program might destroy part of another's program in memory, automatic memory protection is provided.

The Type 166 processor contains two 9-bit registers, the protection register and the relocation register. When the system is in the protection mode, the contents of the protection register are compared with the high order nine bits of each memory address requested by the program. If any bits have ones in both the protection register and the memory address, an error is signaled and the memory cycle does not take place. If no error occurs, the address sent to the memory system consists of the original nine low order bits and the bit by bit inclusive OR of the contents of the relocation register and the high order nine bits of the requested address.

Before running a program in the protection mode, the protection register and relocation register are loaded by the executive program.

MULTIPLE PROCESSOR SYSTEMS

There are many possible systems in which more than one processor uses the same memory module or modules. Two examples of multiple processor systems are described below. In all such cases, the memory module bus interfaces are assigned priority levels: one interface has the highest priority, another the second highest, and the remaining two interfaces are alternately assigned the third highest priority.

Figure 12 illustrates a system in which two arithmetic processors share one memory module. A typical use of this configuration would be where incoming data was to be processed by A, using information stored in modules 1 and 2, then transferred to B for final processing in connection with information stored in modules 4 and 5.

Since a PDP-6 processor can be any device that has the proper interface to its memory bus, a drum processor can be combined in the system with an arithmetic processor. This combination is especially effective in time-sharing applications where core memory capacity may be limited. In Figure 13, one user's program is operating in the arithmetic processor and memory modules 1 and 3, while another user's program is being loaded into module 2.
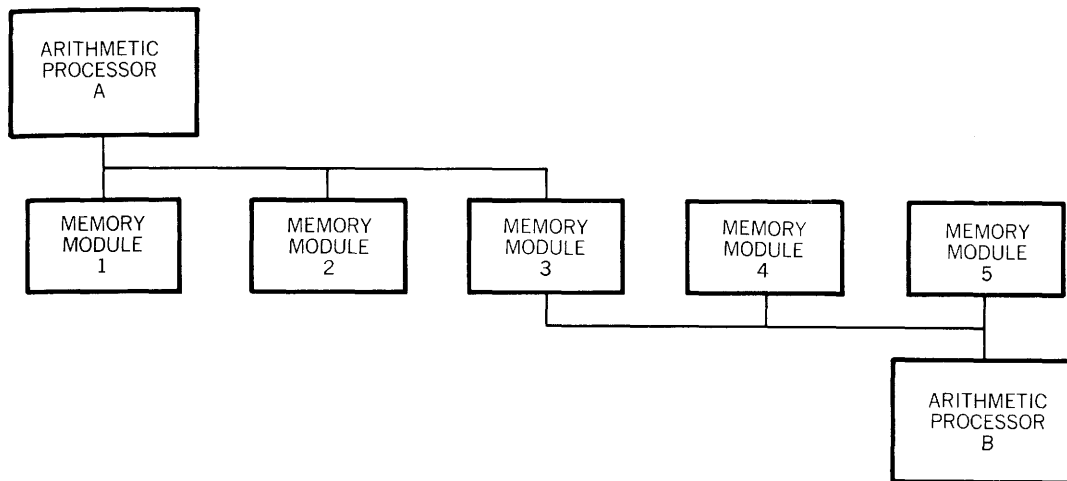
Figure 12   Partial Memory Sharing by Two Arithmetic Processors
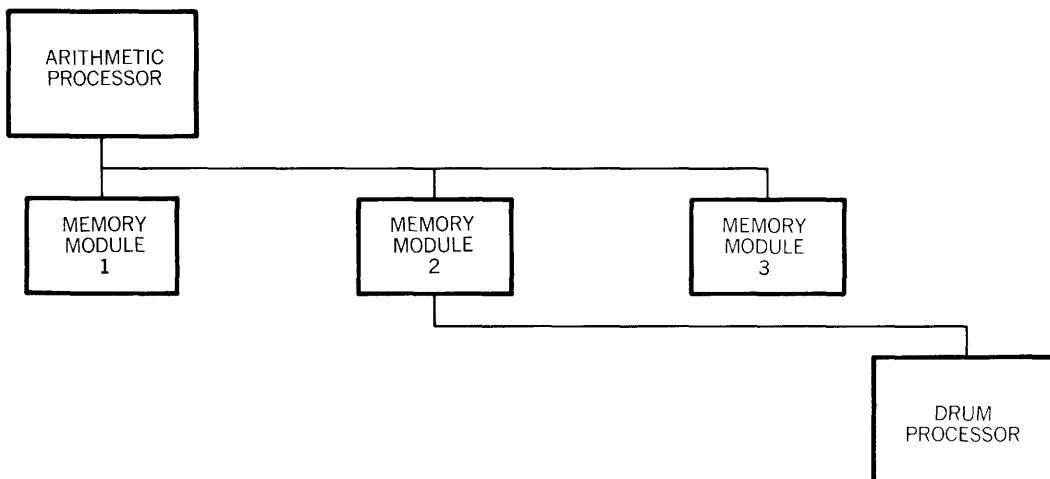
Figure 13   Use of Backup Storage in Time-Sharing Operations

The address selection schemes used by the processors may be entirely independent. Both processors may address the memory modules in the same way; one processor may address module 1 as the low memory field and module 2 as the high field, while the other processor addresses them in reverse order. One processor may have address overlapping while the other does not. In short, each memory-processor system functions as if there were no other system sharing its components. However, in programming the two systems to function simultaneously due regard must be given to the relationship between the address selection schemes.

It is impossible to illustrate all of the memory-processor systems that could be put together. A multiple processor system could conceivably consist of any number of processors and any number of words of directly addressable memory less than 262,144 times the number of processors, with memory modules interconnected in a network-like arrangement. The only restrictions would be that a single processor could not address more than 262,144 words, and that a single module could not be addressed by more than four processors.

Memory Timing

MEMORY MODULE TYPE 163

The Type 163B Core Memory Module contains 8,192 36-bit words, the Type 163C contains 16,384 words. The timing sequence for both is shown in Figure 14[1].

Memory Request	Memory Acknowledge	Data in MB	Start Write Cycle	Memory Ready for New Request
0	0.4	0.8	1.2	2.0
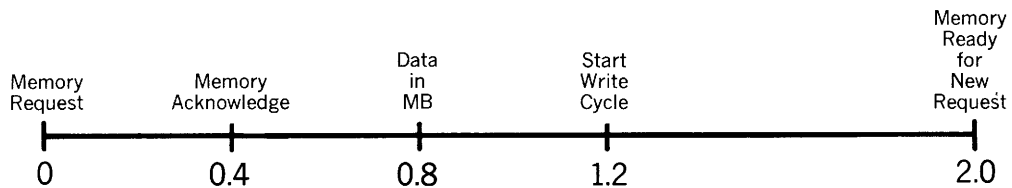
Figure 14   Type 163 Core Memory Internal Timing

Note that the data resulting from a read request is available in the memory buffer at 0.8 microseconds. At that time memory will signal the processor to take the data and proceed with its operation. During the write cycle, the data which has been read out of the specified memory location is written back into the location.

The write only mode of operation is exactly the same as the above except that the processor places data into the buffer register and is dismissed as soon as it receives the acknowledge signal, that is, 0.4μsec. While the memory goes through a read cycle, the contents of the buffer are not changed so that the information placed in it by the processor is written into the memory location during the write cycle.

In the pause mode of operation, the write cycle, shown above as beginning simultaneously with the end of the read cycle, will not begin until a signal is received from the processor.

---

1. Nominal times; actual times may be shorter.

MEMORY MODULE TYPE 162

The Type 162 Memory Module contains 16 36-bit words of flip-flop memory with a cycle time of 0.4 microseconds. Acknowledgement and data are available at the end of this time.

EXECUTION TIMES

The time required for the processor to execute a given instruction must include the time required for all necessary memory references. Consequently, the speed of a PDP-6 system depends on the memory configuration and on how it is utilized. Chapter 2 contains a flow diagram for calculating all instruction times and a table summarizing instruction times assuming various locations of the instruction and data. The following diagram illustrates the reduction of total instruction time that results when instruction, accumulator, and operand are in different memory modules. Assume that the instruction is in core memory, the accumulator in fast memory, the memory operand in a different core memory module than the instruction, and that the result is stored in the accumulator.
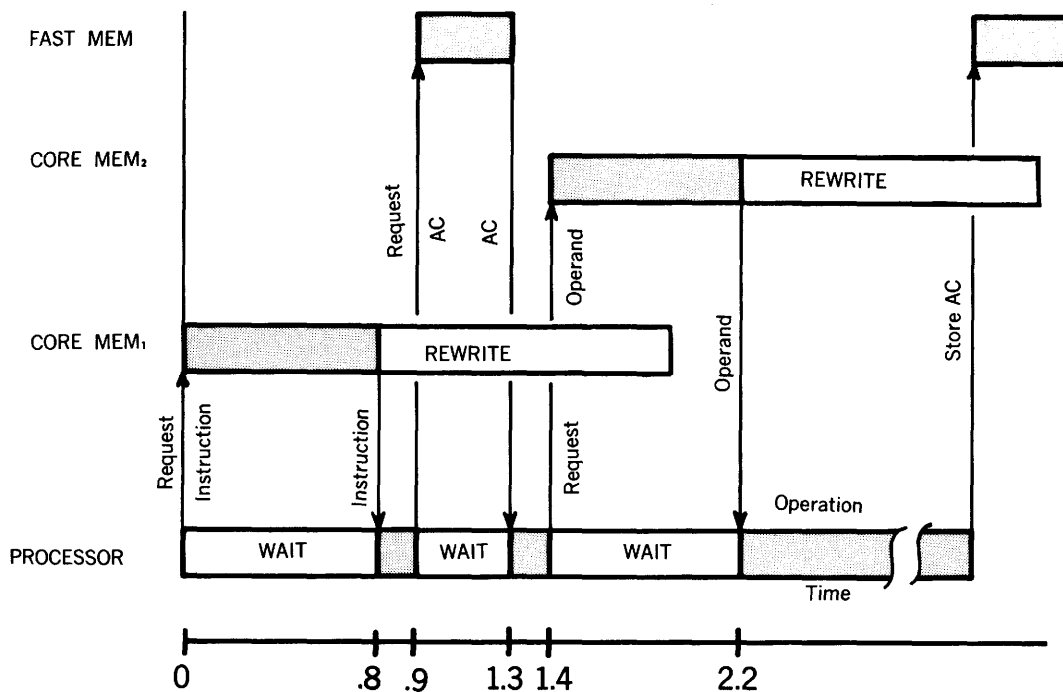
Figure 15   Typical Memory Timing Using Three Memory Modules in the Execution of One Instruction

CRT Display 346
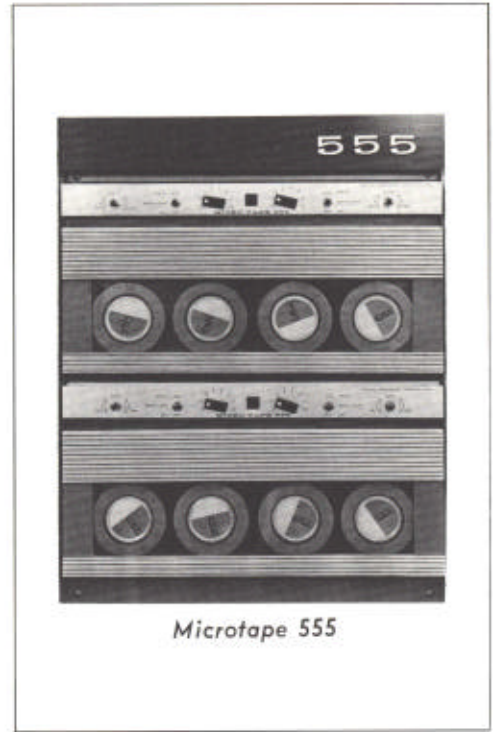
Microtape 555

Line Printer 646
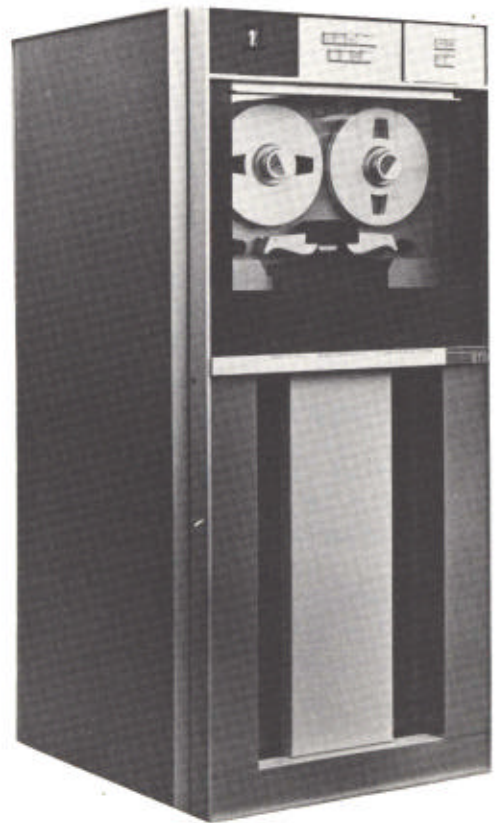
Tape Transport 570

CHAPTER 4
INPUT-OUTPUT

PDP-6 input-output systems consist of up to 128 devices, which may include different types of processors, connected in parallel through a common bus to a single processor. The processor, in addressing any I/O unit, sees only the interface of the single bus. Unit selection is done by placing a device number on separate lines of the bus. Each I/O device control has its own built in logic circuitry to decode its own device number and respond to an I/O command only when its own number is on the lines.

There are only four functions performed by the I/O control units: take initial conditions from the data lines, place status bits on the data lines, take data from the data lines, and place data on the data lines. To carry out these functions, the I/O controls are provided with separate data registers, control registers, and status registers.

The data registers vary in size from 6 to 36 bits. In the case where the data registers are less than 36 bits they are connected to the low order lines; that is, data enters and leaves the processor accumulator in the low order bit position.

The control and status registers vary in size up to 18 bits, and for most units are the same register. The processor is able to specify the function it wishes the device to perform by loading initial conditions into the control register, or it can determine the current state of any device by requesting that it place the contents of the control and/or status register on the lines. Whenever a device has a separate status register in addition to the control register, the status register is assigned an additional device number.

In the PDP-6 there are only four basic I/O instructions corresponding to the above four functions:

Conditions Out (CONO)   Set the control register to specify the operation of the device.

Conditions In (CONI)   Transmit the contents of the control and/or status register to the processor.

Data In (DATAI)   Transmit the contents of the data register of the device to the processor.

Data Out (DATAO)   Transmit the contents of the I/O register in the processor to the data register of the device.

The complete I/O instruction consists of bits to specify one of the above commands along with a 7-bit device number. Four additional instructions apply only to the Type 166 processor, such as skip on given conditions from the status register. They do not issue any new commands to the I/O system.

Priority Interrupt System

Since a device can only take data from the bus, or place data on the bus, upon simultaneously receiving its own device number and a command from the processor, there is no danger of conflicting usage. However, it does raise the question of how a device can signal the processor if it needs immediate attention. For this purpose a seven channel priority interrupt system has been provided. Any number of I/O

devices can be connected to any one of the seven priority interrupt lines. The connection is made whenever a conditions out instruction places the three bit binary number for the channel in the low order three bits of the device control register, zero being equivalent to no connection.

When any of the devices connected to a priority line needs service, indicated by predetermined bits in the cöntrol register being set to one, and the priority line is free, the priority line begins requesting a program interrupt. The processor will honor the request, with commands to the devices known to be on the line, if there are no higher priority channels (channels with a lower number) in an interrupt, and if it is not committed to finish its current instruction.

Input-Output Devices

The input-output devices are listed with their device numbers in numerical order. Included under each device are a brief description and control word configuration. The essential information, such as the control registers, status registers, and operation codes, is summarized in a table at the end of this chapter.

PERFORATED TAPE PUNCH TYPE 761

Device Number: 100_8 (001 000 0)

Mnemonic: PTP

The Type 761 is a Teletype BRPE punch which perforates 8-hole tape at 63.3 characters (lines) a second. The following figure shows the timing for the punch.

*Determined By Punch

BUFFER MUST BE LOADED BY THIS TIME TO ALLOW 63.3 CHAR/SEC OPERATION
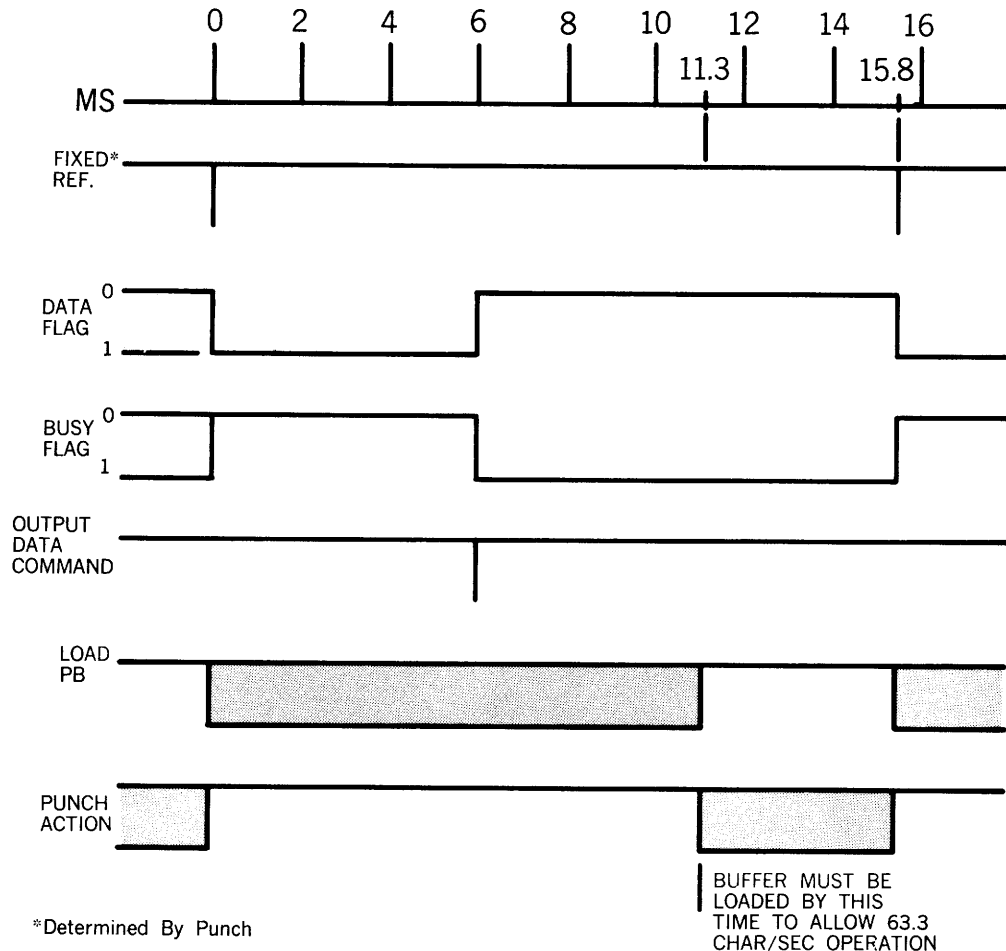
The punch can operate in one of two modes, alpha or binary. In the alpha mode, the eight bits of the data register are punched in all eight holes with ones only punching holes. In the binary mode, the eighth hole is automatically punched and the seventh hole punch is suppressed. The low order six bits in the data register are punched in the remaining holes.

Since the punch data register is only eight bits, information to be punched should be located in the low order eight bits of the I/O registers of the processor.

An automatic timing switch prevents punching for one second after the first setting of the Busy flag to allow the motor to reach punching speed. If no data-out commands appear for five seconds, the motor is turned off.

The control register configuration for the perforated-tape punch is given below.
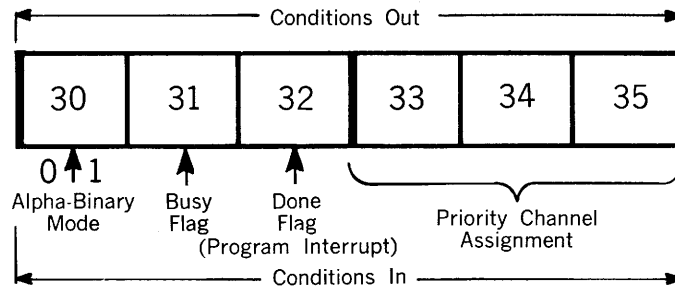
If bit 30 is 0, the punch is in the alpha mode; if a 1, the punch is in the binary mode.

To program, use a conditions-out instruction to set the mode bit and the Busy flag to start the motor, followed by a data-out command. Additional data-out commands may be given when the Done flag is raised. The data-out commands will clear the Done flag and set the Busy flag.

PERFORATED TAPE READER TYPE 760

Device Number: 104ₓ (001 000 1)

Mnemonic: PTR

Depending on the guide setting, the perforated-tape reader is capable of reading photoelectrically 5-, 7-, or 8-hole perforated-paper tape at 400 lines a second. The figure below shows the timing for the reader.
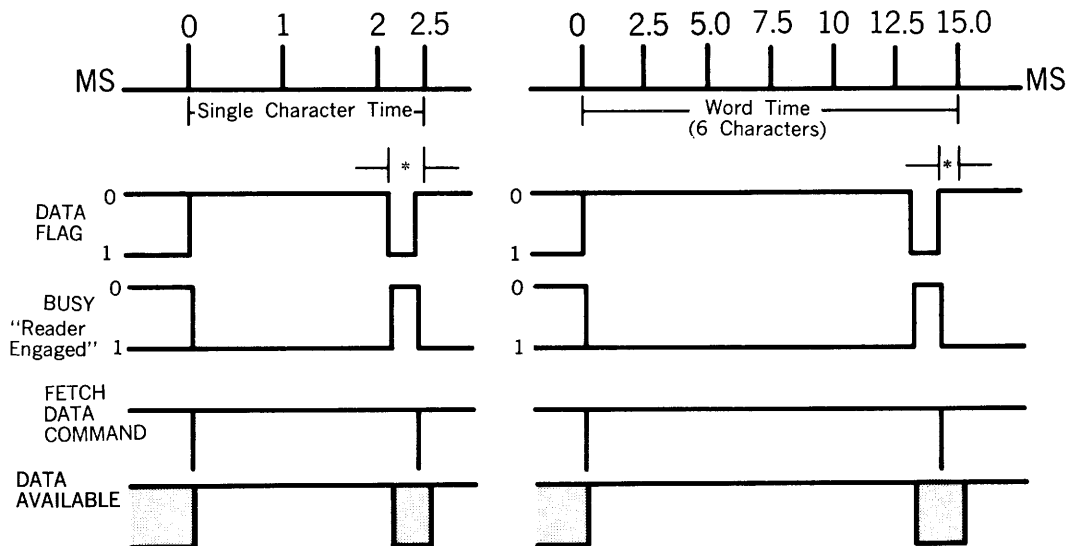
ALPHANUMERIC ON "CHARACTER" MODE

The next "fetch data" command must be given during this 400 microsecond interval to keep the Reader running at maximum rate

The reader can operate in one of two modes, alpha or binary. In the alpha mode, all eight holes are read into the data register, and the device clears the Busy flag and sets the Done flag. In the binary mode, a line of tape is read only if the eighth hole is punched. The seventh hole is ignored and the remaining six are read. Six rows are read for each read command, and the results are assembled into a 36-bit word in the data register. The first line goes into the high order bits, and each succeeding line goes into the next lower order position. When the word is complete, the device again clears the Busy flag and sets the Done flag.

The control register configuration for the perforated tape reader is given below.
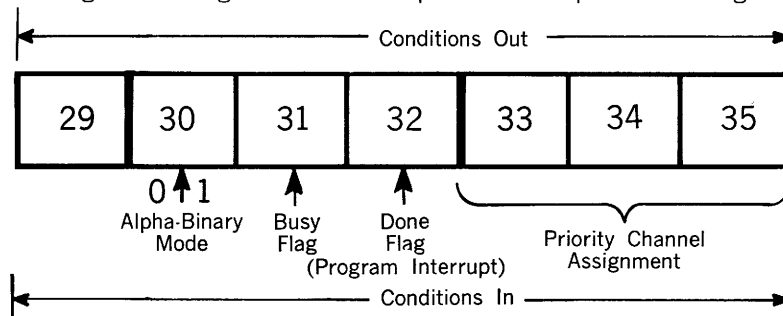
If bit 30 is 0, the alpha mode is specified; if a 1, the binary mode is specified. To instruct the reader to read one data word, set bit 30 to 0 or 1 and bit 31 to 1 with a conditions-out instruction. When the data-in command is given. the Busy flag is set to 1, and the Done flag is cleared causing the reader to get the next character.

CARD PUNCH CONTROL TYPE 460

Device Number: 110_8 (001 001 0)

Mnemonic: CP

The Card Punch Control Type 460 permits punching standard IBM cards at 100 cards a minute. Following is the control register configuration for the card punch control.
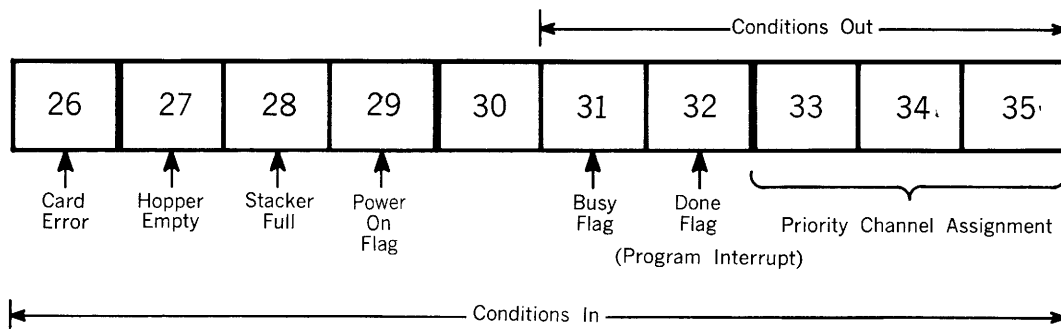
To program the card punch, 80 words are transmitted 12 times a card. The first time the words are transmitted, bit 24 of each word is loaded in sequence into an 80-bit buffer by the card reader control module. Then row 12 is punched. The next time the 80 words are transmitted, bit 25 of each word is loaded into the buffer to control punching for row 11. This process is repeated with bits 26-35 until rows 0-9 have been punched. A 1 in any bit means punch a hole.

To begin punching a card, a conditions-out instruction should be used to clear the Done flag and set the Busy flag. Setting the Busy flag will move a card into punch position. The Busy flag will then remain set until the entire card has been punched.

When the card is in position, the Done flag is set to request one word of information. The data-out command transmits a word of information and clears the Done flag. When the control module has finished reading the appropriate bit into the buffer, the Done flag is set to read the next word. When all 80 words have been transmitted, the row is punched before the Done flag is set to request again the first word of the sequence. Finally, when all 12 rows have been punched, the Busy flag is cleared by the device.

CARD READER TYPE 461

Device Number: 114 (001 001 1)

Mnemonic: CR

The Type 461 Card Reader reads standard 80 column cards at 200 cards per minute. Cards are read on a column by column basis.

The card reader reads in one of two modes: Hollerith or column binary. In the Hollerith mode the 12 bits of a column are converted to a 6-bit code. The column binary mode transmits the 12 bits of a column in unaltered form. In the Hollerith mode six 6-bit characters are packed in a word, and in the binary mode three 12-bit characters are packed in a word.

In the Hollerith mode the first word is composed of columns 1 through 6, the second of 7 through 12, the third of 13 through 18, etc. for a total of 13 words. Fig. 16 shows the reader card code. In the column binary mode the first word is composed of columns 1 through 3, the second of 4 through 6, the third of 7 through 9, etc. for a total of $26\frac{2}{3}$ words.

The control register for the card reader has the following layout:
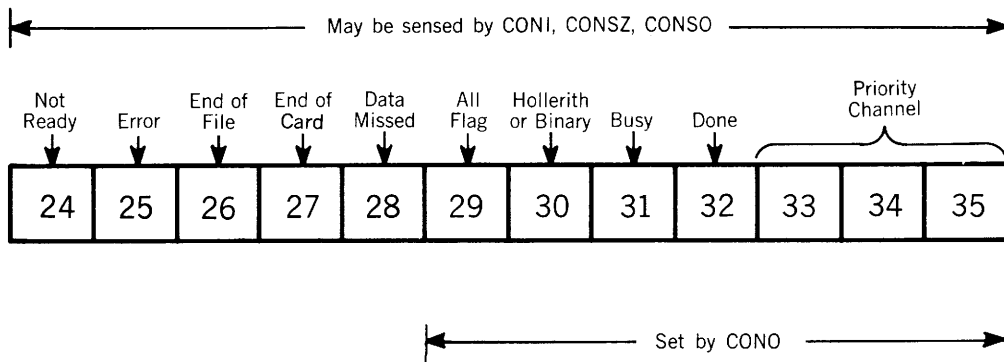
The significance of the bits of the status register is as follows:

24   Not Ready flag: Set to 1 if the power is off, validity check error (validity checking must be on), a card jam, card stacker is full, card hopper is empty, covers not in place, START button not depressed, or a read circuit failure.

25   Error flag: Is set to 1 if there is a failure in the card reader.

26   End of File: Is set to 1 when the END OF FILE button on the card reader is depressed to indicate that all cards have been read.

27   End of Card: Set to 1 when all 80 columns have been read. This bit sets the Done flag and clears the Busy flag.

28 Data Missed: Set to 1 if information in the buffer register is not read before the arrival of new data.

29 All flag: If set to a 1, the Done flag is raised after reading every column.

30 Mode: If a 0, the reader is in Hollerith mode;
      If a 1, the reader is in column binary mode.

31 Busy flag: Set to 1 if the reader is in the process of reading a card.

32 Done flag: Set to 1 whenever a full word has been assembled unless the All flag is set, or is at the end of the card.

33-35 Priority channel assignment.

Card Code		Internal Code	Character	Card Code		Internal Code	Character
Zone	Num.			Zone	Num.		
—	—	01 0000	Blank	11	0	10 1010	↑
12	8-3	11 1011	.	11	1	10 0001	J
12	8-4	11 1100	)	11	2	10 0010	K
12	8-5	11 1101	]	11	3	10 0011	L
12	8-6	11 1110	<	11	4	10 0100	M
12	8-7	11 1111	←	11	5	10 0101	N
12	—	11 0000	+	11	6	10 0110	O
11	8-3	10 1011	$	11	7	10 0111	P
11	8-4	10 1100	*	11	8	10 1000	Q
11	8-5	10 1101	[	11	9	10 1001	R
11	8-6	10 1110	>	0	8-2	01 1010	;
11	8-7	10 1111	&	0	2	01 0010	S
11	—	10 0000	—	0	3	01 0011	T
0	1	01 0001	/	0	4	01 0100	U
0	8-3	01 1011	,	0	5	01 0101	V
0	8-4	01 1100	(	0	6	01 0110	W
0	8-5	01 1101	"	0	7	01 0111	X
0	8-6	01 1110	#	0	8	01 1000	Y
0	8-7	01 1111	%	0	9	01 1001	Z
—	8-3	00 1011	=	—	0	00 1010	0
—	8-4	00 1100	@	—	1	00 0001	1
—	8-5	00 1101	↑	—	2	00 0010	2
—	8-6	00 1110	'	—	3	00 0011	3
—	8-7	00 1111	＼	—	4	00 0100	4
12	0	11 1010	?	—	5	00 0101	5
12	1	11 0001	A	—	6	00 0110	6
12	2	11 0010	B	—	7	00 0111	7
12	3	11 0011	C	—	8	00 1000	8
12	4	11 0100	D	—	9	00 1001	9
12	5	11 0101	E	All other codes		00 0000	←
12	6	11 0110	F				
12	7	11 0111	G				
12	8	11 1000	H				
12	9	11 1001	I				

Fig. 16. Card Reader Codes

To read a card, the "card reader busy" bit is set to 1, the mode is specified, and the priority interrupt channel assignment is made. Once the CONO command is given, the reader will read all 80 columns in order. The end-of-card bit becomes a 1 after the last column is read. The end-of-card bit in turn, sets the Done flag and clears the Busy flag. At this time, columns 79 and 80 are read, and are in the buffer right-adjusted.
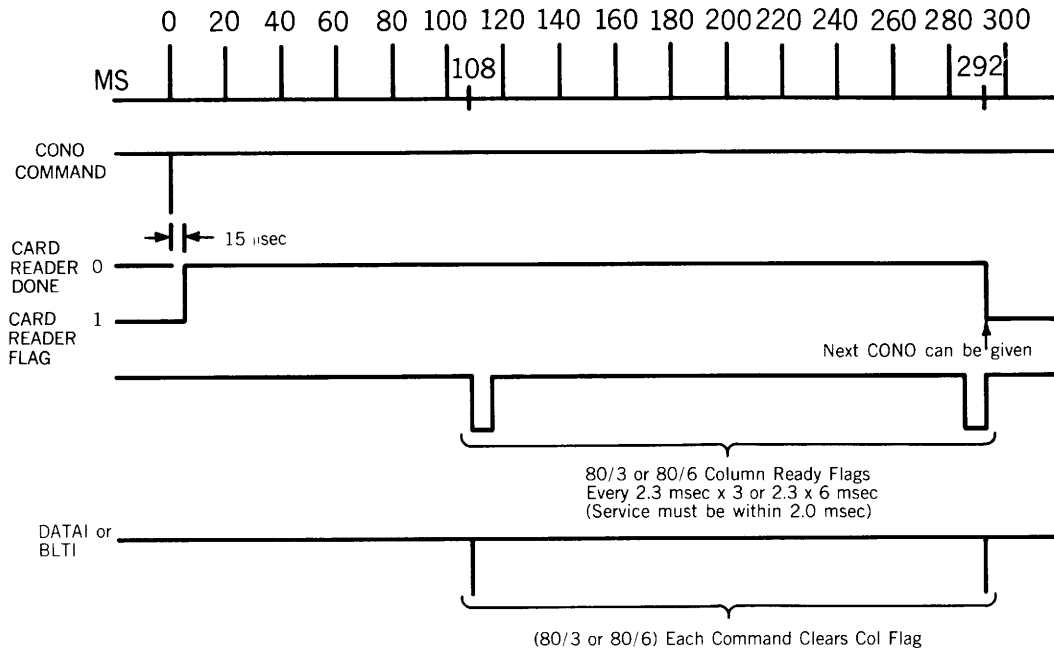
Card Reader Timing

If it is desirable to read a deck with both Hollerith and column binary cards, this may be done by setting the All flag to a 1 for the first column and the mode to column binary. The presence of a 7 and 9 punch indicates a binary card. Otherwise, the card is in Hollerith. A new CONO instruction may then be given. The instruction will clear the buffer register, and if given within 2 milliseconds, the first column of the card will be reread.

TELEPRINTER TYPE 626

Device Number: 120_8 (001 010 0)

Mnemonic: TTY

The Printer-Keyboard is a console Teletype Model KSR 33 which can print or receive ten characters a second as represented in Figure 17 by the Teletype eight-level code.

The signals to and from the KSR to the control unit are standard serial 11 unit-code Teletype signals. The signals are: start (1 unit), eight signals dependent upon the eight information bits (1 unit each), and stop (2 units). The figure below indicates the current pattern produced by the binary code 11010110.
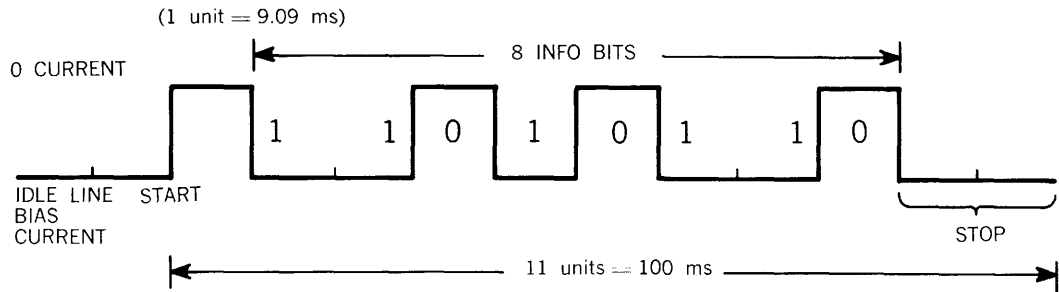
TELETYPE TIMING OF INFORMATION CODE 110 10 110

The control register configuration for the printer-keyboard is given below.
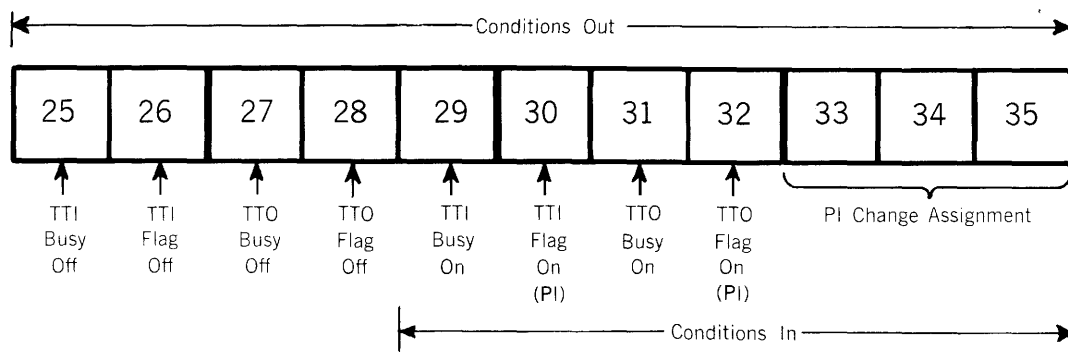
Since the teleprinter consists of two independent devices, a keyboard and a printer, the control register must provide independent setting of TTI and TTO bits. Consequently, this control register functions differently from those of other devices. Only bits 33–35 can be cleared by a conditions-out instruction transmitting zeros. A conditions-out instruction can only set the remaining bits by transmitting ones; zeros are ignored. If, for example, the TTI flag is 1, the only way it can be cleared is by a data-in instruction or by a conditions-out instruction setting bit 26 to a 1.

Striking a key clears the TTI flag and sets the TTI Busy flag. When all eight bits of the code are available in the data register, which is eight bits long, the TTI Busy is cleared and the TTI flag is set. Execution of a data-in command clears the TTI flag. Note that since the keyboard and printer function as independent devices, striking a key does not cause printing. Printing follows typing only if a program transmits the input character back to the printer.

1 = HOLE PUNCHED = MARK
0 = NO HOLE PUNCHED = SPACE

MOST SIGNIFICANT BIT
LEAST SIGNIFICANT BIT
8 7 6 5 4 S 3 2 1

	@ col	SPACE col	Function	8	7	6	5	4	S	3	2	1
	@	SPACE	NULL/IDLE				0	0		0	0	0
	A	!	START OF MESSAGE				0	0		0	0	1
	B	"	END OF ADDRESS				0	0		0	1	0
	C	#	END OF MESSAGE				0	0		0	1	1
	D	$	END OF TRANSMISSION				0	0		1	0	0
	E	%	WHO ARE YOU				0	0		1	0	1
	F	&	ARE YOU				0	0		1	1	0
	G	'	BELL				0	0		1	1	1
	H	(	FORMAT EFFECTOR				0	1		0	0	0
	I	)	HORIZONTAL TAB				0	1		0	0	1
	J	*	LINE FEED				0	1		0	1	0
	K	+	VERTICAL TAB				0	1		0	1	1
	L	,	FORM FEED				0	1		1	0	0
	M	—	CARRIAGE RETURN				0	1		1	0	1
	N	.	SHIFT OUT				0	1		1	1	0
	O	/	SHIFT IN				0	1		1	1	1
	P	0	DCO				1	0		0	0	0
	Q	1	READER ON				1	0		0	0	1
	R	2	TAPE (AUX ON)				1	0		0	1	0
	S	3	READER OFF				1	0		0	1	1
	T	4	(AUX OFF)				1	0		1	0	0
	U	5	ERROR				1	0		1	0	1
	V	6	SYNCHRONOUS IDLE				1	0		1	1	0
	W	7	LOGICAL END OF MEDIA				1	0		1	1	1
	X	8	S 0				1	1		0	0	0
	Y	9	S 1				1	1		0	0	1
	Z	:	S 2				1	1		0	1	0
	[	;	S 3				1	1		0	1	1
	\	<	S 4				1	1		1	0	0
	]	=	S 5				1	1		1	0	1
	↑	>	S 6				1	1		1	1	0
RUB OUT	←	?	S 7				1	1		1	1	1

8	7	6	
1	0	0	SAME
1	0	1	SAME
1	1	0	SAME
1	1	1	SAME

FIGURE 17. Teletype Eight Level Code

The figure below illustrates the keyboard timing; that is, the time required for the input to be assembled and available in the data register from the time the key is struck.

COMPUTING TIME = 78.8 MS
BETWEEN CHARACTERS
AT MAXIMUM RATE
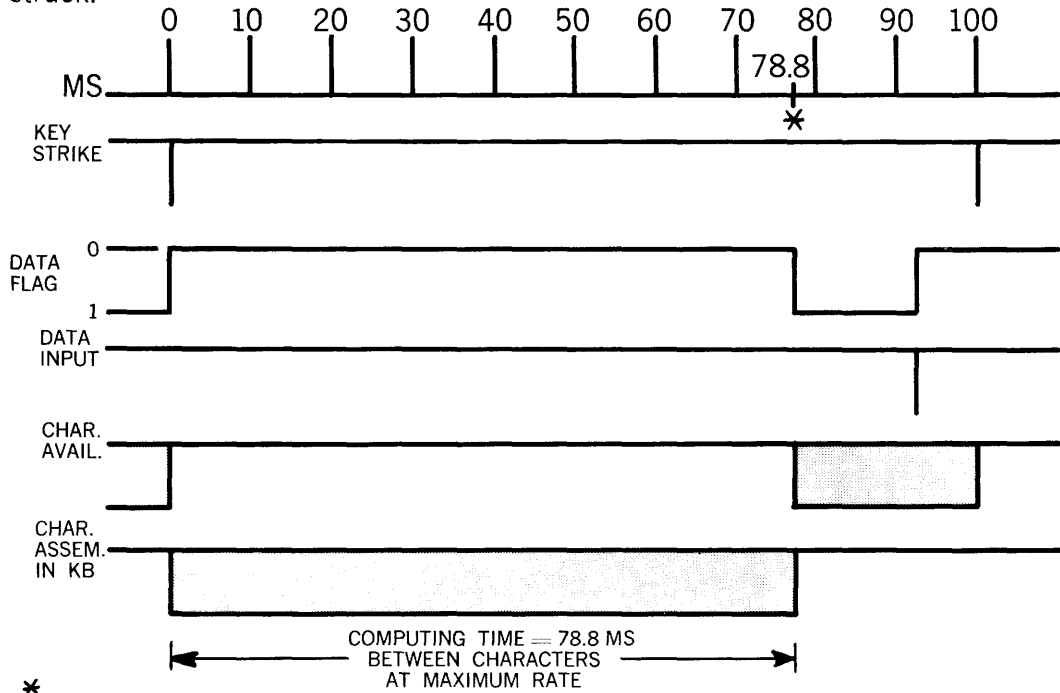
✳ Information Available at Middle of 8th Information Bit

For printing, a data-out command places an 8-bit code in the data register, clears the TTO flag, and sets the TTO Busy flag. When the character has been printed, the device clears the TTO busy bit and sets the TTO flag. The following figure shows the timing for the printer.

BUFFER MUST BE LOADED BY THIS TIME
TO ALLOW 10 CHAR/SEC OPERATION
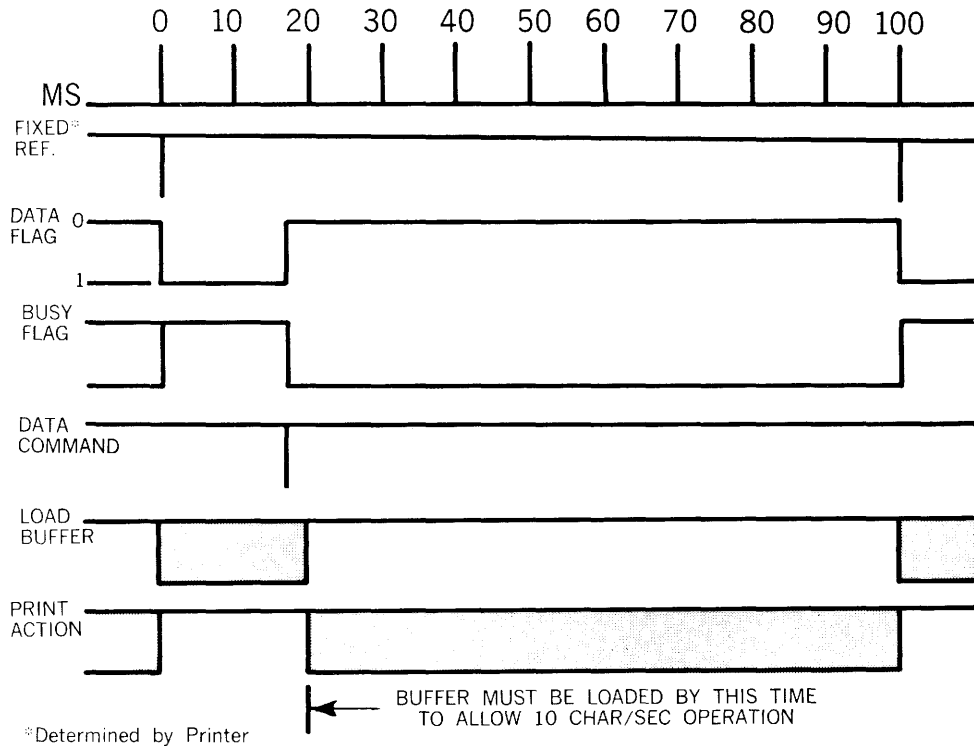
✳Determined by Printer

LINE PRINTERS TYPE 646

Device Number: 124_8 (001 010 1)

Mnemonic: LPT

Three on-line printers are available: Type 646A operates at a maximum speed of 300 lines per minute; Type 646B at 600 lpm; and Type 646C at 1000 lpm. All use the same control. The control register configuration is:

```
|<----------------- Conditions Out ----------------->|
| 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
   ^    ^    ^   _____/     _____/
   |    |    |
 Error Busy Done   Priority Channel    Priority Channel
 (PI)       (PI)    for Error Flag       for Done Flag

|<----------------- Conditions In ------------------>|
```
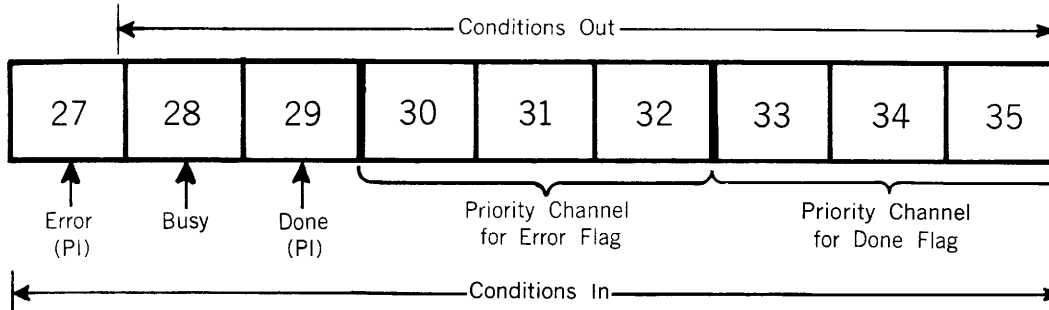
The printer system contains a core buffer capable of holding up to 120 characters. Execution of a conditions-out command clears the data register, clears the control register and sets the priority channel assignments, and initiates clearing of the printer buffer. The conditions-out instruction automatically sets the Busy flag to a 1 and clears the Done flag. When the buffer is clear, the Busy flag is reset to 0, and the Done flag is set to 1. The Error flag is set if any condition exists which would make the printer inoperable.

When the Done flag is on (1), five 7-bit ASCII coded characters in bits 0–34 may be transferred to the printer data register by a data-out command. The data-out command automatically clears the Done flag and sets the Busy flag. The five characters are processed from left to right and sent to the printer buffer. When all five characters have been sent to the buffer, the Done flag is reset and the Busy flag is cleared. No more than 120 characters should be transferred without giving a print control character, as defined below.

The line printer control characters are a subset of the Type 626 Teleprinter control characters. All other teleprinter control characters are ignored by the printer; that is, an illegal code is simply shifted out of the data register and the next character, if legal, is sent to the printer. The legal control characters are:

HORIZONTAL TAB (011) — A horizontal tab of 8, 10, 12, 16, or 20 columns can be selected by means of a rotary switch mounted in the printer buffer. Upon receipt of this character, space characters are loaded into the printer buffer until the tab stop position in memory is reached. The next character sent to the buffer is printed in the column corresponding to the tab stop.

FORM FEED (014) — prints and spaces paper to top of the next page. The line of characters previously stored in the buffer is printed and paper motion started. Track 1 of the vertical format tape is selected (see below). The column counter is not reset; so the first character printed on the next page appears in the column to the right of the last character before the form feed.

CARRIAGE RETURN (015) — Prints out the line of characters stored in the buffer and resets column counter. Paper is not spaced.

EOT (004) — This character is used when there is no more printing to be done. It clears the Busy flag without setting the Done flag.

VERTICAL FORMAT TAPE — Whenever paper spacing occurs, the printer spaces until a hole is seen in the specified tape channel. Any carriage control tape may be placed in the printer, but a tape in the following format is supplied as standard:

Control Character	Track	No. of Lines to Next Hole on Standard Carriage Control Tape
Form Feed (014)	1	Skips to top of page
DC_0 (020)	2	1
DC_1 (021)	3	2
DC_2 (022)	4	3
DC_3 (023)	5	6
DC_4 (024)	6	11
Vertical Tab (013)	7	22
Line Feed (012)	8	33

In addition four blank lines are on the control tape before the line which has a hole in channel 1 (top of form). This insures that the bottom two and top two lines of each page are always left blank. This leaves 62 printing lines per page.

INCREMENTAL CRT DISPLAY TYPE 346

Device Number: 130_8 (001 011 0)

Mnemonic: DIS

The Type 346 is an incremental display with point plotting, line generator, vector, and character generator modes. The display must be used with the interface for PDP-6. Display Monitor Type 343, a slave display, can be used in addition to the Type 346 to provide additional cathode ray tube displays for multiple consoles.

The control register configuration is shown below.
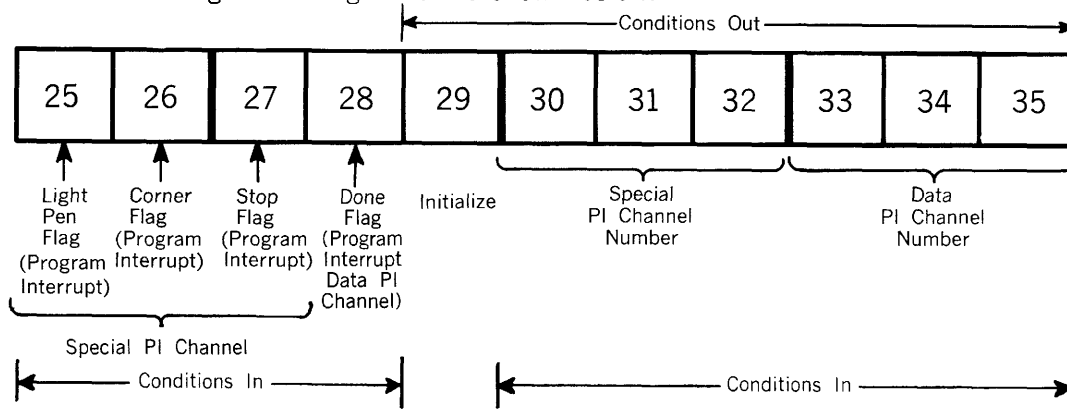
Bits 25-28 cause an interrupt. However, unlike most devices for the PDP-6, two priority channels may be assigned. The Done flag, bit 28, requests an interrupt on the data PI channel; the remaining interrupt flags request an interrupt on the special PI channel.

The Light Pen flag is set by the pen whenever it is in use and detects a spot on the scope. The use of the Edge flag and the Stop flag will be explained in the discussion that follows.

Whenever any interrupt occurs on the special PI channel, the programmer has the option of resuming operation with the existing conditions, or he may initialize the display by setting bit 29 to a 1 with a conditions-out instruction.

The Type 346 display is controlled by 18-bit control words, which are packed two to a PDP-6 memory word. The parameter and X-Y formats are shown below.
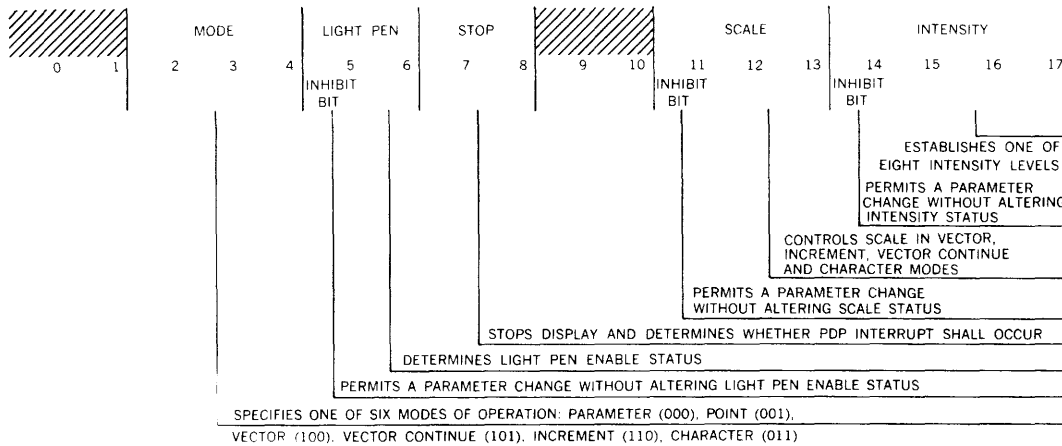
PARAMETER MODE
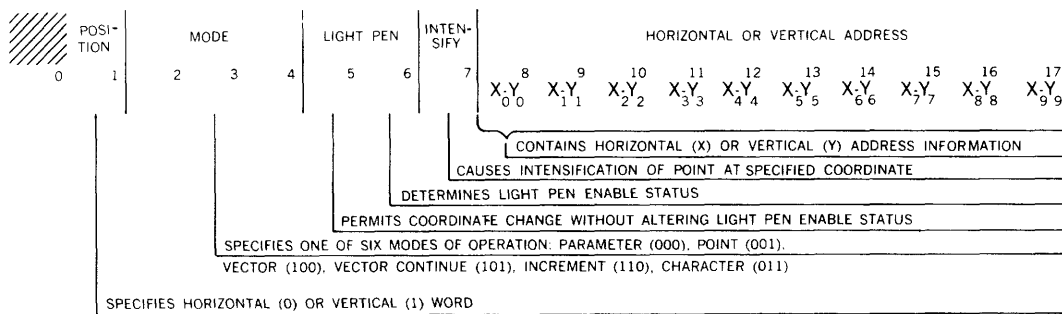
POINT MODE

In the X-Y mode the horizontal and vertical coordinates are identified by a 0 or a 1 respectively in bit one. All data is transmitted to the display as 36-bit words; the display control examines the left 18 bits first, then the right 18 bits. The order in which the X-Y words appear is not critical; however, when both horizontal and vertical words must be used to set up a single plot, faster operation is achieved by having the vertical word in the left half.

The bits in the X-Y mode have the following significance:

Bit 0   Presently unused.

Bit 1   Axis index. Must be a 0 to indicate a horizontal coordinate, a 1 to indicate a vertical coordinate.

Bits 2, 3, 4 These specify the mode of operation for the subsequent data. These bits are stored so that all data words following a single control word are interpreted in the mode specified by the previous control word. An escape mechanism is provided in each mode in order that the display can be forced to return to the control word mode for a change in position, parameter, or mode.

Bit 5   This bit enables a change in the light pen enable status, bit 6.

Bit 6   Subject to bit 5 being a one, this bit enables the light pen when a 1, disables the light pen when a 0.

Bit 7   Intensify; if a 1, a single point will be plotted at the specified coordinates.

Bits 8-17   These bits specify the horizontal or vertical coordinate, subject to the axis index, bit 1.

The bits of the parameter mode format are interpreted by the display as follows:

Bits 0	Presently unused
Bit 1	Presently unused
Bits 2-4	Mode bits which have the same significance as bits 2-4 of the X-Y mode.
Bits 5, 6	Light pen, with the same significance as bits 5, 6 cf X-Y mode above.
Bit 7	Stop bit; if a one, the data request is inhibited and the display halts.
Bit 8	Interrupt on stop. Only if this bit is a 1 will the display interrupt the programming device on a stop.
Bits 9, 10	Presently unused.
Bit 11	If a zero this bit inhibits the setting of the scale bits, 12 and 13.
Bits 12, 13	These two bits control the size of the character in the character mode or the scale in the vector and increment modes.
Bit 14	If a zero this bit inhibits the setting of a new intensity level.
Bits 15, 16, 17	Used to select one of the possible intensity levels.

POINT PLOTTING MODE — This mode can be used to plot individual points located at random on the tube face. Whenever a new point is specified by the control word, there is a set-up delay of 30 microseconds.

THE INCREMENT MODE — In the increment mode an 18-bit data word, whose format is shown below, will cause the plotting of four successive points. These words are packed two to a PDP-6 word and are transmitted as 36 bits to the display. The display interprets the left 18 bits first followed by the second 18 bits.
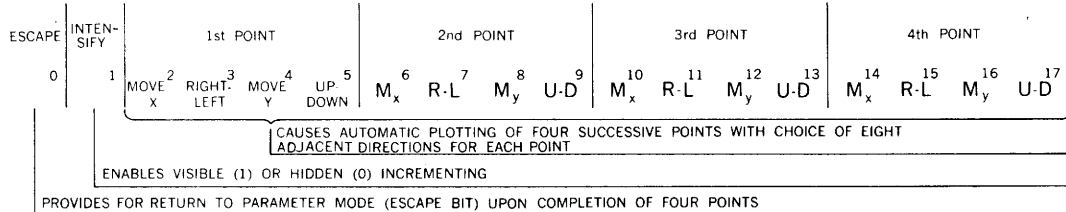
INCREMENT MODE

Each point is specified with four bits which describe the direction of motion about the current spot location. With four bits, one of eight possible points immediately adjacent to the present spot location can be plotted. Each point requires 1.5-μsec plotting time. Four zeros in an increment character inhibit motion in either direction, and no point will be plotted. To increment the spot without intensifying, the intensify bit, bit 1, is made a 0.

Once the display has been programmed to the increment mode, all incoming data is interpreted as increment mode data. Escape from the increment mode is accomplished by making bit 0 a 1. When the escape bit has been enabled, the four specified points are plotted, and the display returns to the parameter mode prior to the next incoming data cycle.

The vector continue mode enables the drawing of straight lines from any given point within the matrix to the edge of the screen with a single instruction. Any direction subject to the maximum (or minimum) delta-Y delta-X ratio can be achieved. An escape to the parameter mode and a request for data automatically result from a vector continue mode cycle.

When the scale bit is enabled, the X and Y registers are incremented by 2^n ($n = C$ (SC_0, SC_1)) whenever a point is plotted. This has the effect of double spacing the points.

An Edge Flag signal is provided whenever the edge of the screen has been violated. An illegal Edge flag will stop the display while inhibiting the data request; a legal Edge flag, for example, at the end of a continue cycle, will not inhibit the data request. In either case, escape to the parameter word mode automatically occurs. The flag is reset when new data arrives.

THE VECTOR GENERATOR — The vector generator used in the vector mode provides a means for displaying straight lines between two points without specifying any in-between points. The vector mode data word, whose format is shown below, consists of eight bits of delta-X information, eight bits of delta-Y information, an intensity bit, and an escape bit. These words are packed two to a PDP-6 word and are transmitted as 36 bits to the display. The display interprets the left 18 bits first followed by the second 18 bits.

ESCAPE	INTEN-SIFY	Δ Y								Δ X							
0	1	$\pm$ SIGN BIT 2	3	4	5	6	7	8	9	$\pm$ SIGN BIT 10	11	12	13	14	15	16	17

Δ Y MAGNITUDE BITS Δ X MAGNITUDE BITS

ENABLES VISIBLE (1) OR HIDDEN (0) VECTORING

PROVIDES FOR AUTOMATIC RETURN TO PARAMETER MODE (ESCAPE BIT) UPON COMPLETION OF PLOTTING
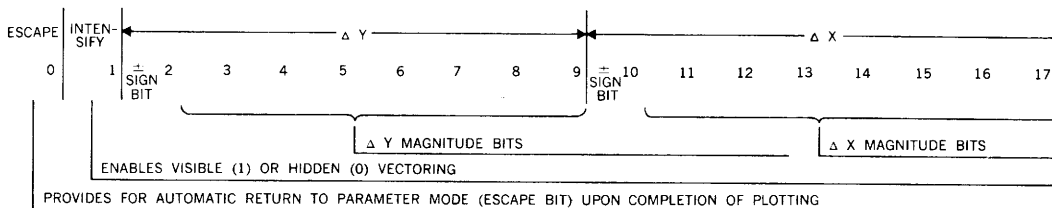
VECTOR MODE

Delta-X and delta-Y each comprise seven magnitude bits and one direction bit. Since the display area consists of a 1024 x 1024 point matrix, the maximum length vector which can be drawn with a single instruction is $\pm$ one-eighth of the display width. There is no limitation for a minimum length vector. Each point requires 1.5-microseconds plotting time.

The escape and intensity bits are used the same in the vector mode as in the increment mode.

THE CHARACTER GENERATOR TYPE 342 — The character generator provides a display of characters. Each character or symbol will be encoded with 6 bits, packed 6 to a 36-bit word. Of a total of 128 characters, the alphabet contains 123. Two characters are used in a pair of case-shift codes to achieve the required encoding capability. Upon specific character-like commands, the character generator also is capable of performing the space and carriage return functions which utilize two more characters. Escape from the character mode is accomplished with a character-like code and accounts for the 128th character.

DATA CONTROL TYPE 136

Device Number: 200

Mnemonic: DC

The Type 136 Data Control is an intermediate device between the processor and other external devices such as tape controls. The 136 serves as a data buffer stage and as a control device relieving the processor of exercising detailed control over the transfer of each individual transfer of data to and from the external device.

As a data buffer, the 136 assembles units of data from the external device, where the units may be 6, 12, 18, or 36 bits, into single 36-bit words for transfer to the processor. Similarly, it accepts 36-bit words from the processor and transfers them to the external device in groups of 6, 12, 18, or 36 bits.

As a control device, the 136 is capable of controlling up to 6 devices, one at a time. The 136 sends and receives all signals to and from the device necessary to control and synchronize the flow of data between the device and the buffer register. Also, it signals the processor when the buffer register is ready to send or receive a full 36-bit word. Figure 18 shows the logical flow of information through the 136.
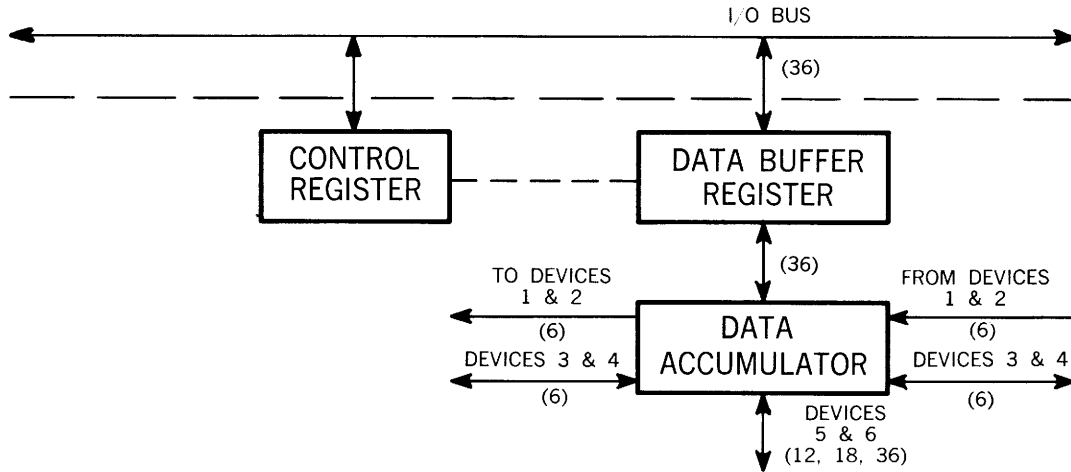
Figure 18 Data Control Type 136

For output, information is transferred from the I/O bus to the data buffer register where it is held until the data accumulator is ready for a new word. The information is then automatically transferred to the data accumulator and the data buffer Request flag is set to get more data from the processor. The data in the data accumulator is then sent to the desired device.

For input, information is transferred from the device to the data accumulator where it is assembled into a complete 36-bit word. It is then transferred to the data buffer register for transfer to the processor via the I/O Bus.

There are three ways in which data can leave the accumulator for the device, or enter the accumulator from the device. Which way is chosen depends on the device being used.

Devices 1 and 2 are 6-bit character oriented devices which can be run in two directions (microtape). Upon receipt of a Take/Give Character Left signal from the device, either six bits of data leave bits 0-5 of the accumulator for the device, if output, or six bits of data from the device enter bits 30-35 of the accumulator, if input, and the accumulator is shifted left six places. If the device were running in the opposite direction, it would issue a Take/Give Character Right signal in which case a similar sequence of events would follow. However, data would leave the accumulator from bits 30-35 and enter bits 0-5 followed by shifting the accumulator right. When all six characters have been shifted into, or out of, the accumulator, the accumulator Request flag is set to request a transfer of data between the accumulator and the data buffer register.

Devices 3 and 4 are 6-bit character oriented devices which can be run in one direction only (magnetic tape). They are exactly like devices 1 and 2 except for being unidirectional; they issue Take/Give Character Left signals only.

Devices 5 and 6 are unidirectional devices as are 3 and 4. However, they may be 12-, 18-, or 36-bit devices. Following is the control register configuration for the data control.

| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |

Set by conditions-out inst

Number of characters Assembled 1
Data Missed
Buffer Accum-ulator Move
Data Accum-ulator Request
Data Buffer Request
In/ Out
Packing Mode
Device Number
Priority Channel

Readable by conditions-in and conditions-skip instructions
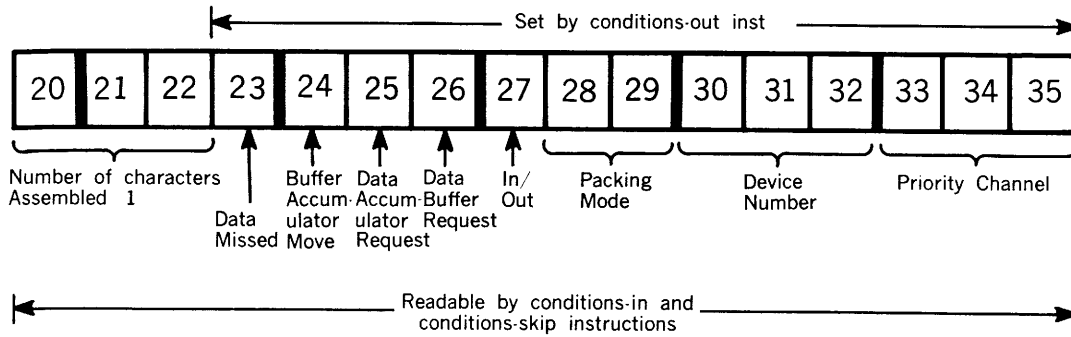
Bit Assignments:

20-22   The number of characters assembled or remaining in the data accumulator minus one.

23   Missed Data. This flag is set to one if the external device attempts to initiate a transfer of data out of or into the data accumulator before the Data Accumulator Request flag is cleared.

24   Buffer-Accumulator Move flag is a synchronization flag for the movement of data between the data accumulator and the data buffer. Data is moved when both the Move flag and the Data Accumulator Request flag are equal to one.

25   Data Accumulator Request flag is set to one when the correct number of characters have been assembled into the accumulator or shifted out of the accumulator.

26   Data Buffer Request flag is set to one when data is available in the buffer to be moved to the processor or when the buffer is ready to accept data from the processor.

27   In/Out   If 0, data is to be transferred from the external device to the processor. If 1, data is to be transferred from the processor to the device.

28-29   Packing Mode
        00   For six 6-bit units of data
        01   For one 36-bit unit of data
        10   For three 12-bit units of data
        11   For two 18-bit units of data

30-32   Device Numbers 1-6.

33-35   Priority Channel Assignment

For output, the Buffer Request and the Data Accumulator Request flags should be set to ones and the Move flag to zero. A 1 in the Buffer Request flag will request the transfer of data from the processor, either by causing a priority interrupt or by being tested by a conditions-in instruction. When the data is set in the buffer, the Request flag is cleared resulting in the setting of the Move flag.

Since the Accumulator Request flag and the Move flag are both 1, the data is moved from the buffer register to the accumulator, the Data Accumulator Request flag and the Move flags are then cleared and the Buffer flag is set to 1, loading the buffer with a second word of data and again setting the Move flag.

The above sequence of events can be occurring while the external device is being made ready to receive data, for example, a tape unit being brought up to speed. As soon as the external unit has requested all the information from the data accumulator, the Accumulator Request flag is set and the above cycle is repeated.

For input, a conditions-out instruction is used to set the Move flag to 1 and the two Request flags to 0. As soon as the external device has requested the transfer of sufficient data to fill the accumulator, the Data Accumulator flag is set to 1. Since the Move flag was previously set to a 1, data is then moved to the Buffer Register, the Accumulator Request flag is cleared to accept the next unit of data from the external device, and the Move flag is cancelled. When the contents of the Buffer Register have been transferred to the processor, the Buffer Request flag is cleared and the Move flag is reset.

The data control may be used in or out of the priority interrupt. In the latter case, data flow is synchronized by examining the status bits with a conditions-skip instruction.

Microtape

Microtape is a bidirectional tape system in which data is written on, and read from, the tape in accordance with a prerecorded block format. Five tracks of information are recorded redundantly making a total of ten tracks on the tape. Two of the tracks are used as a timing track and a mark track. The timing track is used to synchronize reading and writing with each line of information on the tape appearing adjacent to a timing mark. The mark track is used to identify the type of information to be found in the three data tracks.

Figure 19 shows the sequence and types of data on the tapes.

Only the data words and block marks are transmitted to the PDP-6. The remaining words on the tape are used as control and error detection information by the Microtape control.

The blocks may contain any number of data words depending on the prerecorded formats. Indeed, each block may contain a different number of data words. However, block lengths of 128_{10} are considered to be standard format for the PDP-6.

Before Microtape is used for data storage it must be prerecorded. Prerecording is done in two passes. During the first pass, the timing and mark tracks are placed on the tape. During the second pass, forward and reverse block mark numbers, the standard data pattern, and the automatic check sums are written. Since part of the data word must be reserved to produce the mark track, it is impossible to write intelligent data in the information channels at the same time. For this reason, the two passes are required. Once these two passes have been made, the tape is said to be prerecorded and is ready for normal use.

The Microtape system includes a programmed mode of operation called write timing and mark track and a manual switch which permits wiring on the timing and mark tracks and also activates a clock which produces the timing track and flags for program control. Unless both the mode and the switch are used simultaneously, it is physically impossible to write on the mark or timing tracks. A red indicator lights on all transports connected to the appropriate control when the manual switch is in the "on" position. In this mode only, information channel 1 (bits 0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33) is also connected to the mark track channel.

The actual mark track which is written on the tape (see Figure 19) was selected after careful consideration and provides many functions, such as:

   a) Program synchronization
   b) End of block detection
   c) Error checking and prevention
   d) Protection of control information
   e) Block and word addressability
   f) Automatic bidirectional compatibility

FIGURE 19 MARK AND INFORMATION TRACK FORMATS

69

g) End of tape detection
h) Variable block format
i) Inclusion of marks to allow expansion for various word lengths.

One other unique feature of the mark track is that the six control marks before the data marks are "complement obverses" of the six control marks after the data marks.* The data mark is the complement obverse of itself. Thus, when reading in the reverse direction the mark track window **sees exactly the same thing in both directions** since the flux reversals on the tape are opposite to those when reading forward and the bits are read in the reverse order. No special logic is required to distinguish the format of the tape in either direction.

The end marks on either end of the tape illustrate this bidirectional ability even better. As the end marks are complement obverses of each other, only that end of tape will be recognized at which the tape will physically come off the reel if further movement continues. Thus, no special hardware is needed for opposite ends of the tape, and there is no harm in coasting into or turning around in the end zones. Errors will be indicated only if attempting to go further into the end zone. The particular bit structure of the end marks is a repetitive one, so that any shift of three bits in the window will appear as another end mark. This makes it virtually impossible to pull the tape off the reel in any of the normal modes.

DUAL MICROTAPE TRANSPORT TYPE 555

The Type 555 Transport consists of two logically independent bidirectional tape drives capable of handling 260 foot reels of ¾ inch, 1.0 mil Mylar tape. The bits are recorded at a density of 375 ($\pm$60) bits per track inch. Since the tape moves at a speed of 80 inches per second, the effective information transfer rate is 90,000 bits per second, or one 36-bit word every 400 microseconds. Traverse time for a reel of tape is approximately 40 seconds.

The 3½ inch reels are loaded simply by pressing onto the hub, bringing the loose end of the tape across the tape head, attaching it to the take up reel and spinning a few times. Individual controls on the transport enable the user to manipulate the tape in either direction manually. The units can be "dialed" into a particular selection address.

There is no capstan or pinch-roller arrangement on the transport, and movement of the tape is accomplished by increasing the voltage (and thereby the torque) on one motor, while decreasing it on the other. Braking is accomplished by a torque pulse applied to the trailing motor. Start and stop time average 0.15-0.2 seconds and turn around time takes approximately 0.3 seconds.

RECORDING TECHNIQUE

The Microtape system uses the Manchester type polarity sensed (or phase modulated) recording technique. This differs from other standard types of tape recording, where for example, a flux reversal might be placed on the tape every time a one is desired. In the polarity sensed scheme a flux reversal of a particular direction indicates a zero while a flux reversal in the opposite direction indicates a one. A timing track, recorded separately in quadrature phase, is used to strobe the data tracks. Thus, the **polarity** of the signal at strobe time indicates the presence of a zero or one. Using the timing track on the tape as the strobe also negates the problems caused by variations in the speed of the tape.

---

*The complement obverse of a word is defined as the complement of a word with the bits read in the reverse direction, e.g.: 010110 (26) and 100101 (45), 001000 (10), and 111011 (73).

With this type of recording only the polarity, not the amplitude of the signal, need be considered, thus removing some of the signal to noise problems and allowing the use of read amplifiers with high uncontrolled gain. This recording also allows the changing of individual bits on the tape without changing the adjacent bits.

Reliability is further increased by redundantly recording all five of the information tracks on the tape. Figure 20 shows the placement of this track. This is accomplished by simply wiring the two heads for each information track in series. On reading, the analog sum of the two heads is used to detect the correct value of the bit. Therefore, a bit cannot be misread until the noise on the tape is sufficient to **change the polarity** of the **sum** of the signals being read. Noise which reduces the amplitude would have no effect.

MICROTAPE CONTROL TYPE 551

Device Numbers:
    210 for Control Register
    214 for Status Register

Mnemonics:
    UTC for Control Register
    UTS for Status Register

The Type 551 Microtape control contains all of the read and write circuitry as well as the circuitry for block detection logic and motion control logic. The 551 is capable of controlling four dual microtape transports or eight tapes. Unlike most devices which take their commands and data from the I/O bus directly, the Microtape controls work in conjunction with the Type 136 Data Control. Therefore, the 551 is controlled by both the control register and its status with respect to the data control.

Data transfers between the 551 and the 136 are on a 6-bit character basis. The data is assembled in the Data Control to form a full 36 bit word before being transferred to the PDP-6.

There are four modes of operation which require that the user either provide information to the microtape system or accept information from the microtape system. These are governed by the Microtape Control Type 551 and are:
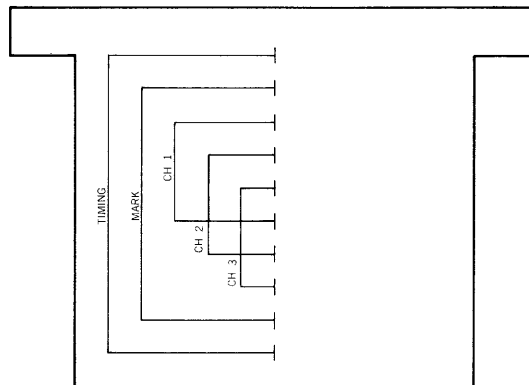
Figure 20   Placement of Microtape Tracks

1. Read or write data mode (Sum check is automatically read and written).
2. Read or write block mark number.
3. Read or write as a continuous record beginning at next reverse block mark.
4. Move tape and do not transfer data.

Figure 21 illustrates the use of the mark track for synchronization and the times at which commands can be given to the data control relative to mark track information.
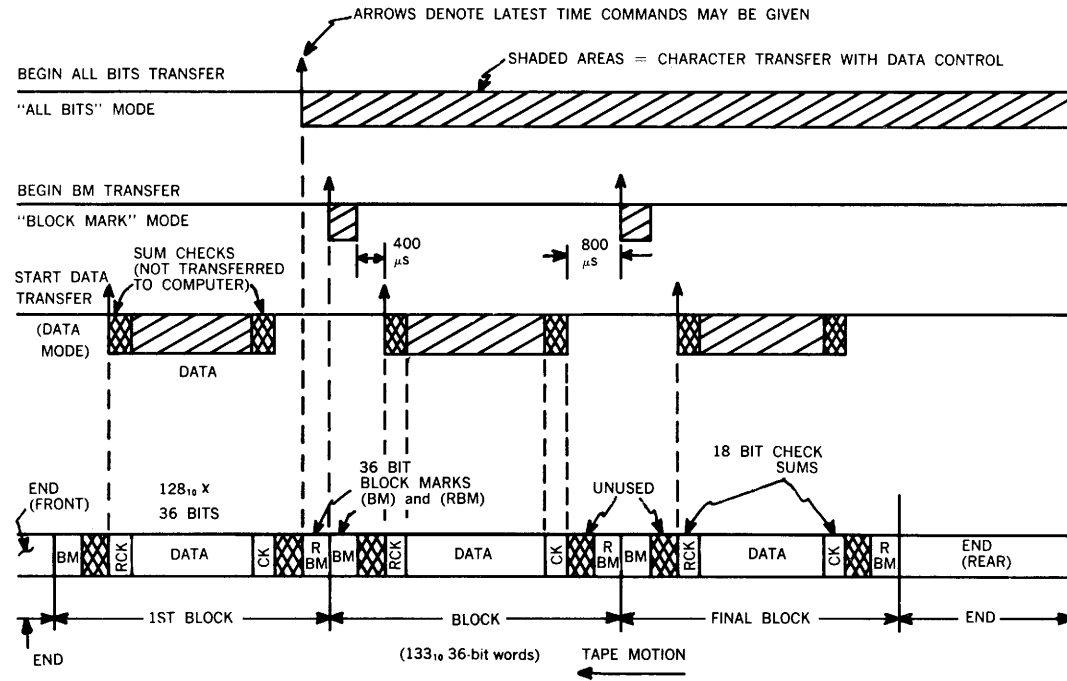
Figure 21   Microtape Format

In order to synchronize the control, the mark track is read by passing the bits through a 6-bit 'moving window' which shifts bit by bit as the tape moves. A decoder associated with the window interprets the pattern present, and takes appropriate action within the control.

The control register configuration is shown below. This register is set by a CONO with device #210 or read by a conditions-in instruction with device #210.
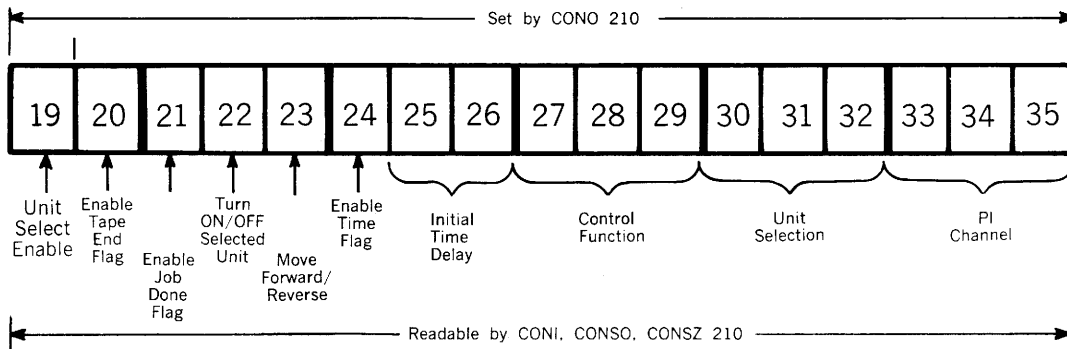
The bit assignments are as follows:

33-35   Sets the priority interrupt channel (1-7; 0 = no interrupt).

30-32   Unit selection (units 1-7 selected by 1-7, unit 8 selected by 0).

27-29   Tape Control function encoded:

000 Select unit and motion. No data transmission.

001 Read all tape as a single record, beginning with the next reverse block mark found.

010 Read block number. Gives a 36-bit block mark number and moves to job done status.

011 Read data and move to job done status if the data control is no longer accepting data or is deselected from the Microtape control.

100 Write timing and mark track as a single, continuous block. (The mark track is a copy of channel 0 on tape.) The WTM ENABLE switch must be "on" within the control.

101 Write all tapes as a single record beginning with the next reverse block mark found.

110 Write Block Number (36 bits) and move to job done status.

111 Write data and move to job done status at end of block if there is no more data to transmit from the data control.

25-26 Specifies 0, 20, 160, or 300 msec. Wait before transmitting any data, sets the Time flag at the end of 20, 160, or 300 msec.

25	26		
0	0	0	msec
0	1	20	msec
1	0	160	msec
1	1	300	msec

24 Interrupt enable the Time flag.

23 Move forward/move reverse. 0 = forward, 1 = reverse.

22 STOP/START the selected unit. 0 = stop, 1 = start.

21 Interrupt enable the Job Done flag.

20 Interrupt the Tape End flag.

19 Unit select enable. This bit must be a 1 to select a unit; if 0, all units are deselected.

Below is the configuration for the status register. This register is cleared whenever a CONO, 210, is given. It may be read by a conditions-in instruction with device number 214.

```
|◄──────────────────── CLEARED by CONO 210 ────────────────────►|
```

| Delay In Progress | Test Conditions | | | Incomplete Block Transfer Flag | Writing ON | Time Flag | Parity Error | Illegal Operation Flag | Tape End Flag | Job Done Flag |
	Request	Active	Null							
25	26	27	28	29	30	31	32	33	34	35

```
|◄──────────────────── Readable by CONI    214 ────────────────────►|
                                   CONSO
                                   CONSZ
```
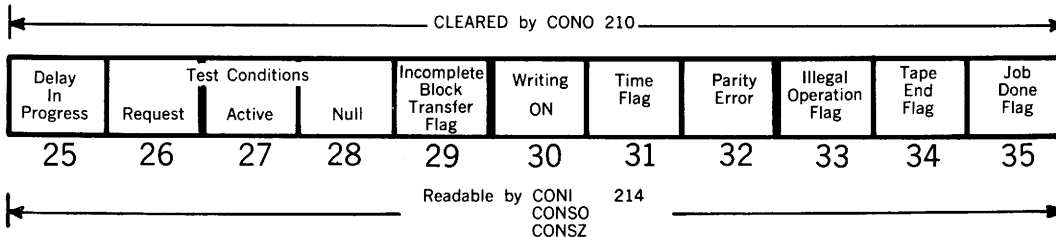
35 Job Done flag — signifies a task has been completed, and may be used to interrupt the program if Job Done Enable is a 1.

34 Tape End flag — Set to a 1 when the tape moves into its end zone and may be used to interrupt if the Tape End Enable is a 1. When the Tape End flag is set, the unit control flip-flop is turned "off".

33   Illegal Operation flag — Set if a write command is given, and a reel is write protected. Also set if no units or more than one unit is selected. The flag causes an interrupt.

32   Parity Error — Set if an error occurs.

31   Time flag — Set by the completion of the specified time delay and may be used to interrupt if the Time Enable is a 1.

30   Writing ON — A test condition which is a 1 during writing operations.

29   Incomplete Block Transfer flag — Set if a complete block has not been transferred, either by deselection of data control from Micro Tape control or data control is receiving or transmitting no more data.

28-27-26   Test conditions which specify state of the control. The control is in the null, active, or request state.

25   Delay in progress (test condition).

Figure 22 shows the operation of the Microtape control.
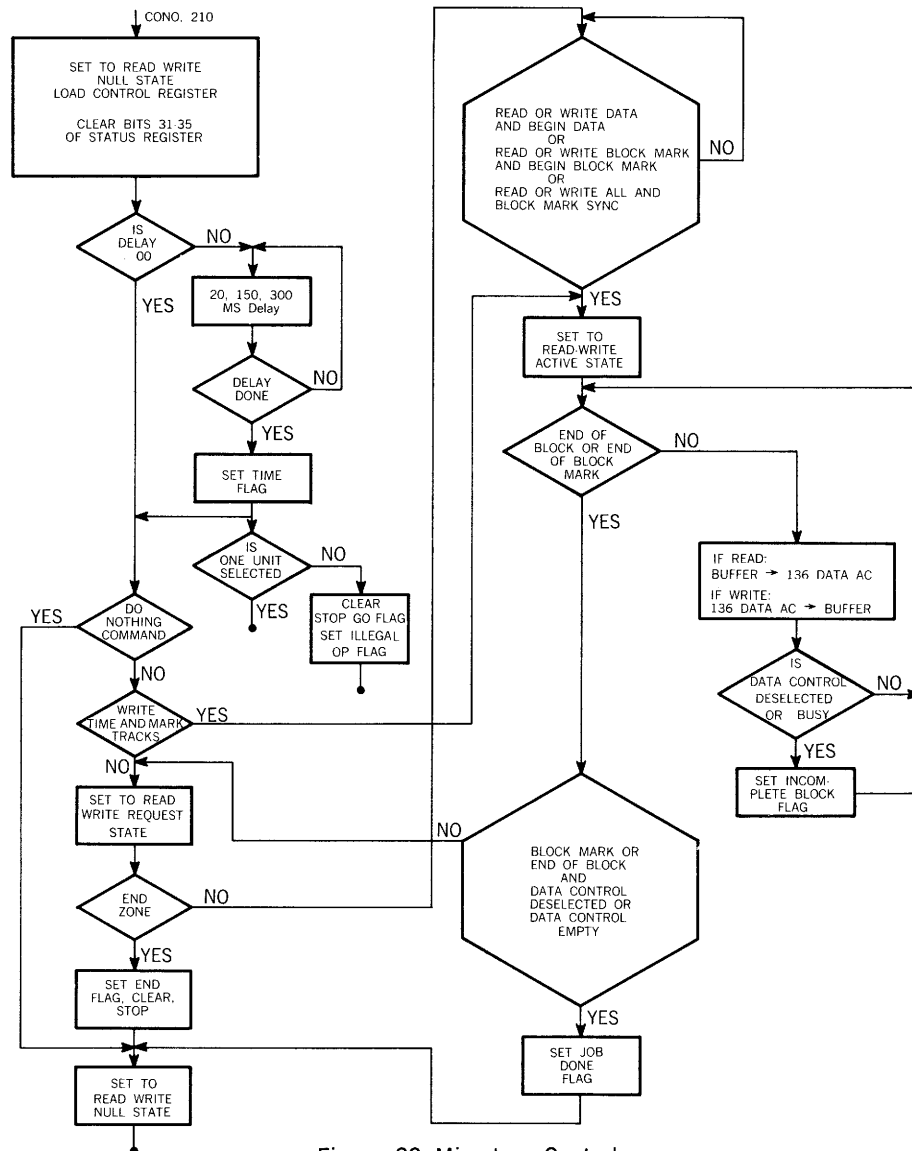
Figure 22 Microtape Control

MAGNETIC TAPE CONTROL TYPE 516

Device Numbers:
 220 For Control Register
 224, 230 For Status Registers

Mnemonics:
 MT For Control Registers
 MTS For Status Registers 224
 MTM For Status Registers 230

The 516 operates with the 136 Data Control and 520, 521, or 522 transport interfaces. Operating through the 520 series interfaces, the 516 will control a maximum of eight transports to provide IBM compatible magnetic tape input or output.

Tapes may be written or read at 200, 556, or 800 densities depending on the transport connected. Tape speeds are 75 inches a second or 112.5 inches a second. At these speeds and densities the time between characters is: 66 μsecs, 24 μsecs, and 16.6 μsecs for the 75 ips speed; 44 μsecs, 16 μsecs, and 11 μsecs, for the 112 ips speed.

All tape functions are transmitted via the PDP-6 I/O Bus to the 516 Control by a conditions-out instruction setting the control register. Functions transmitted to the control are placed in a hold register until the previous function has been completed. If the same function is given before the previous one has been completed, the control acts as if it had been placed in a continue or proceed mode. This mode of operation terminates as soon as a different function is given or the previous command completes before a new function is transmitted.

However, the sequence of commands, read, space, read compare, constitutes an exception. Since these are all read commands in the forward direction, the transport would stay in the continuous mode of operation as long as the command hold register is filled.

If an illegal command is given, it will not be accepted. If an illegal condition arises during the execution of a command, the operation terminates, the illegal Command flag is set, and no more commands of that type are accepted in the continuous mode which is then terminated. The control functions defined are:

REWIND
The selected transport rewinds tape to the load point and stops.

REWIND/UNLOAD
Tape is wound off the take up reel of the selected transport.

WRITE
N characters may be written from consecutive or nonconsecutive locations of memory into one record. Binary (odd) or BCD (even) parity may be selected. When the write function is given, all BCD characters 00_8 will automatically be changed to 12_8. Otherwise, BCD 00_8 would be written as blank tape and could not be detected within a record except for the fact that a Status flag would indicate a missed character read while writing. All characters are read and checked for parity while writing. The LPCC (Longitudinal Parity Check Character) that is written is compared when it is read. Separate Status flags are provided for both types of possible parity errors.

WRITE END OF FILE
Write EOF is an automatic instruction and does not require the use of the 136 Data Control. The end-of-file mark is written 17_8 BCD. It is automatically detected during a read or space instruction.

WRITE BLANK TAPE

Writes a full reel of blank tape to end point. To write 3 inches of blank tape, the programmer gives a write EOF command and then a space back command. In either case, the 136 Data Control is not needed.

READ

N characters may be read in either parity mode. When the read function is given, all BCD characters 12_8 will automatically be changed to a 00_8. An additional read buffer is incorporated into the 516 design for generating a longitudinal check sum that is compared with the check character when it is read.

Whenever the specified word count is exceeded in the PDP-6, the 136 Control is disconnected from the 516, but the 516 may not finish its function until ERF is seen and the proper shut down delays have completed. If the word count has not overflowed before the ERF is seen, the transport is shut down unless the command hold register contains another command. If so, the word count may overflow within the next record. This implies gather reading or read continuous.

If a record ends containing a character count that is not a factor of six, the characters appear right-justified in the 136 Data Control. The program must test and read the 136 for the remaining characters.

READ BACKWARD

Identical to read except for the direction of tape motion. Tape recorded at 800 bpi cannot be read backward.

READ COMPARE

N characters are read and compared character by character with words stored in the PDP-6 memory. The 136 Data Control sends characters to the 516 where the comparison takes place. The comparison is performed in the LPCC read buffer. In this case, the LPCC is not read or compared. This function combines the capabilities of the read and write functions. An inequality sets the Read Compare Error Status flag.

SPACE FORWARD
OR BACKWARD

All spacing is done without the use of the 136 Data Control. Spacing forward or backward one record is automatic. If a bit is specified, tape is spaced until a file mark is read. Spacing continuously in one direction is achieved by loading the command hold register before the previous space command has been completed.

Regardless of the function given, the programmer can examine the status register whenever he pleases. A flag called ERF (End of Record) is supplied to the programmer through the 7 channel priority interrupt system. The flag may cause an interrupt to any one of the 7 channels. The flag is also available in the status register.

The control register is set or sensed by an I/O instruction with device number 220_8.

Set by CONO																	
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

False End of Record | Write Clock Enable | Slice Level | Parity | Density | Function | Hold Selected | Unit Selection | Disable EOR Test | Priority Channel

Sensed by Conditions-In
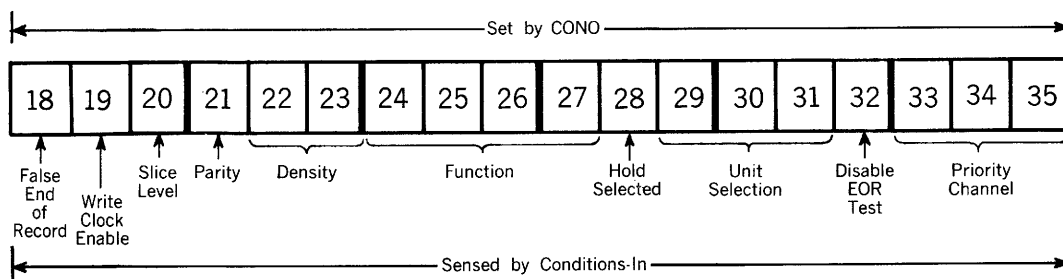
The significance of the control register bits is as follows:

Bits	Description
33-35	Priority channel assignment
32	If a 1, the EOR test is disabled for maintenance purposes. An inconsistent format can be read. Every character on the tape is read.
29-31	Unit selection (0-7)
28	If a 1, do not return transport (Type 570) to pool.
24, 25-27	Tape control functions:

24	25-27	
0	000	No operation
0	001	Rewind selected transport
1	001	Rewind/unload selected transport
0	010	Write at selected transport
0	011	Write end-of-file mark
1	011	Write blank reel
0	100	Read-compare
0	101	Read
1	101	Read backward
0	110	Space forward one record
1	110	Space forward until file mark
0	111	Space backward one record
1	111	Space backward until file mark

22, 23   Density select

22	23	
0	0	200 bpi
0	1	556 bpi
1	0	800 bpi
1	1	556 bpi

Bit	Description
21	Parity If 0, BCD (even) If 1, binary (odd)
20	Slice If 0, high sense level If 1, low sense level
19	Write clock enable
18	False end of record

Bit 28 of the control register requires some additional explanation. The Type 570 Transport may be operated in a pool whereby it can be selected by either one of two controls. Figure 23 illustrates such a pooled arrangement. When a control is finished with the transport, the transport is returned to the pool unless bit 28 is set to 1. In that case the transport can no longer be selected by the other control until some future time when it is returned to the pool.
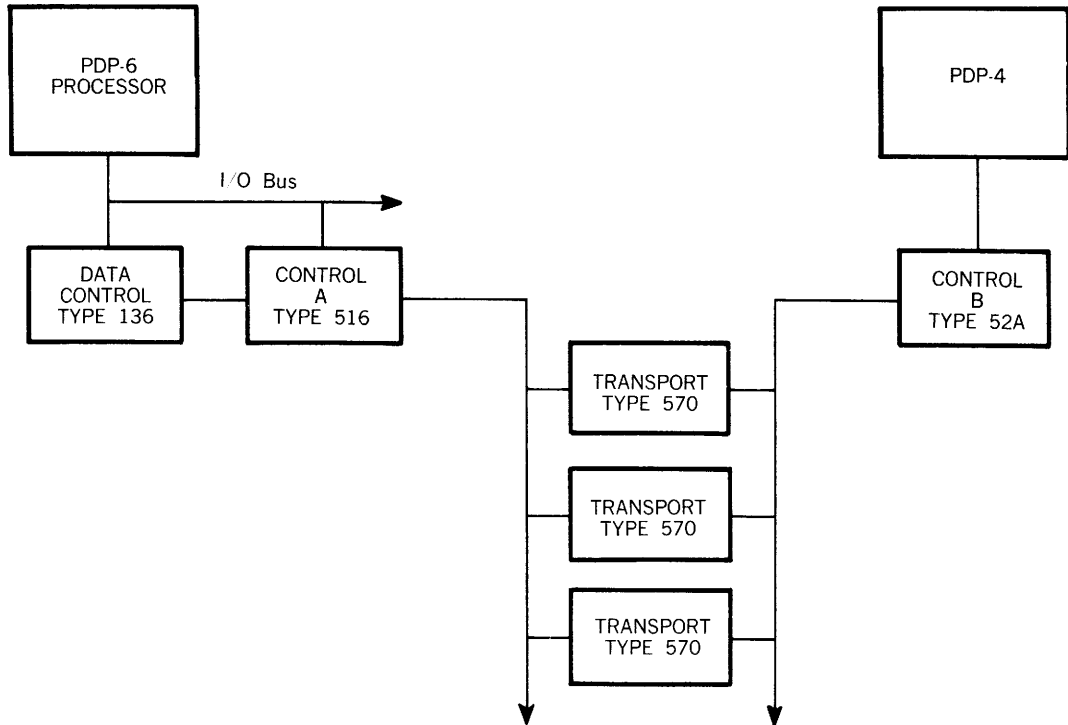
Figure 23   Pooled Transports

The 516 Control contains two status registers in addition to the control register. Neither of these can be set by a CONO instruction; both may be sensed by a conditions in instruction.

The configuration for the status register sensed with device No. 224_8 follows:

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Illegal Flag

Motion Stop Delay In Progress

Transfer New Command to Command Hold Buffer

Continue Motion Selected

Transport Rewinding

Tape at End Point

Tape at Load Point

See Note A Below

See Note B Below

End of File Flag

File Protect Ring Out

Tape Missed Character

Read Compare Error

Tape Parity Error (Loug)

Tape Parity Error (Lat)

End of Record Flag

Tape Transport Ready

Tape Control Free

Note A

Tape near end point. (520 interface)
First operation write (521 or 522 interface)

Note B

Tape near load point if 520 interface
B Control using transport (521 interface) See Pool
Write Echo Not OK (522 interface)

78

Below is the configuration for the status register sensed with device No. 230₈.

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Transport Up to Speed Delay in Progress

End of Record Motion Delay in Progress

Command Hold Buffer Full

Record In Progress

Motion Stop Delay In Progress

Unit Selected New Command

Function Final Buffer

Motion Delay In Progress

Unit Final Buffer

First Char Inhibit

Missed Char Delay

EOR Write Delay

EOR Read Delay

DATA COMMUNICATION SYSTEM TYPE 630

The 630 Data Communication System (DCS) is a real time interface between Teletype stations and PDP-6. It is used for multiuser time sharing systems, message switching systems, and data collection-processing systems. Its basic function is to receive and transmit characters. When receiving, characters of different data rates and unit codes arrive from the Teletype stations in serial form. The DCS converts the signals to Digital voltage levels; the characters are converted from serial to parallel form and are forwarded to the computer. When transmitting, characters in parallel form are presented to the DCS by the computer. The characters are converted to serial Teletype form of the correct data rate and unit code, they are converted from Digital voltage levels to Teletype station signal levels, and they are sent to the Teletype stations.

The modularity and plugability of the 630 DCS simplify the expansion of the system from one station to the maximum number of stations. Various combinations of data rates, unit codes, station types, and station signal level can be accommodated in one 630 DCS.

The 630 System consists of 631 Data Line Interfaces, 632 Send/Receive Groups, and a 633 Flag Scanner. It has a maximum capacity of 8 groups (8 stations per group) or 64 stations (128 pairs of wires for full duplex operation). Figure 24 is a simplified block diagram of the system.

The Type 631 Data Line Interface converts Teletype station signal levels to Digital voltage levels, and converts Digital voltage levels to Teletype station signal levels. The extent of modularity of the 631 is dependent upon the type of station signals to be converted. The 631 is plug connected to the 632.

The Type 632 Send/Receive Group converts parallel characters to serial Teletype characters, or converts serial Teletype characters to parallel characters. It mixes the received characters of the eight Teletype stations onto a bus for presentation to the 633, and notifies the 633 when service is required.

When a character has been received or transmitted, a flag (indicator) is activated. The flag in turn notifies the 633 that service is required for that particular station. The manual OFF-ON switch mounted on the handle of the receiver and transmitter modules may be turned off to inhibit the flag from requesting service.

The Type 632 can accommodate a maximum of eight receiver modules and eight transmitter modules. The quantity required is dependent upon the number of Teletype stations. (If four half duplex stations are to be interfaced, only four receiver and four transmitter modules are required.) The type of each module required is de-

pendent upon the data rate, unit code, and the number of data bits. Teletype stations requiring different data rates, unit codes, and data bits can be intermixed in the Type 632. The receiver module disregards hits (noise) less than one-half of a unit in length on an idle line. The 632 is completely pluggable.

The Type 633 Flag Scanner decodes and interprets computer instructions, forwards received characters to the computer upon request from the computer, sends characters to the transmitter modules when instructed by the computer, scans each 632 in search of activated flags, notifies the computer when an activated flag has been found, and forwards the station number requiring service to the computer upon request from the computer.

The Type 633 contains a precision crystal-controlled clock that generates highly accurate timing pulses. The transmitter and receiver modules use the pulses to sample the serial Teletype signals. An additional crystal clock can be added to accommodate multiple Teletype speeds. A crystal clock is also used to generate timing pulses that control the search logic of the scanner. The scanning mechanism of the 633 is modular. Each expansion permits eight additional stations (1 group) to be scanned.

A rotating priority scanner notifies the computer when an active flag has been found. The computer program requests the station number and then handles the character. Programmed priority of the stations is permitted.

The flag scanner operates at the following speeds: the maximum total time required to examine 64 inactive stations, 32 microseconds; the maximum total time to search, notify the computer and continue to search for 64 simultaneously active stations, 544 microseconds (exclusive of computer interrupt and programming cycles); the minimum time required to find the next active station upon being released by the computer, 6 microseconds; and the maximum time required to find the next active station (station being serviced minus one) upon being released by the computer, 92 microseconds.

PROGRAMMING FOR THE 633
HALF DUPLEX

Device Numbers:
300₈ (011 000 000)

304₈ (011 000 100)

Mnemonics:
DCSA, DCSB

The control register is assigned to device number 300. Following is the control register configuration.

Flag   Priority Channel
                    Assignment
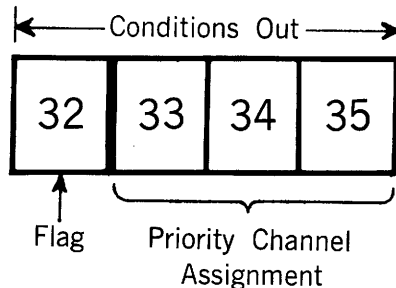
The line number counter and send buffer (used to select a transmitter on an idle line) are assigned to device number 304.

The effect of an I/O instruction depends upon the device number (mnemonic). The various effects may be summarized as follows:

Symbolic Operation	Function
CONO DCSA, E	The priority interrupt channel defined by bits 33-25 of E is assigned to the data communication system. The scanner is released if bit 32 is 1. The counter will be reset to 0 and the scanner will be released if bit 31 is a 1 (programmed priority).
CONI DCSA, E	The previously assigned priority interrupt channel is read into the C(E) bits 33–35. The state of the Scanner flag is read into C(E) bit 32 (1 = on or active). The C(E) bits 0–31 are cleared.
DATAI DCSA, E	The character in the receiver (as specified by the receiver counter) is read into bits 28–35 of C(E). The C(E) bits 0–27 are cleared; the Receiver flag is turned off. The scanner and/or Scanner flag are not affected by this instruction.
DATAO DCSA, E	The transmitter (as specified by the send buffer) is loaded from bits 28–35 of C(E). The Receiver flag is turned off.
CONO DCSB, E	The send buffer is loaded from bits 30–35 of E (used to select a transmitter on an idle line).
CONI DCSB, E	The line number in the receiver counter is read into bits 30–35 of C(E). The C(E) bits 0–29 are cleared.
DATAI DCSB, E	The character in the receiver (as specified by the receiver counter) is read into bits 28–35 of C(E). The C(E) bits 0–27 are cleared. The Receiver flag is turned off, the Scanner flag is turned off and the scanner is released.
DATAO DCSB, E	The transmitter (as specified by the receiver counter) is loaded from bits 28–35 of C(E). The Receiver flag is turned off, the Scanner flag is turned off and the scanner is released.

The method of transmitting is as follows:

1. A send buffer is provided to initiate transmission to an idle station. The program loads the send buffer with a specific station number. The program then instructs the 633 to send the character to the station (using the send buffer to select the desired station). The associated Receiver flag is cleared at this time. Note that the scanner is not released. Thus neither the position of the scanner counter nor the state of the Scanner flag need be considered when initiating transmission on a line. If transmission is initiated in multiple priority levels, it is suggested that the interrupt system be disabled during the setting of the send buffer and sending the character.

2. At the completion of sending a character the Receiver flag is turned on.

3. Steps 1, 2, 3, and 4 for receiving now are executed.

4. At this point, the program has the option of executing the following three steps.

5. The receiver receives all data sent on the telegraph line; therefore, the character received may be used as an echo check on data sent. The telegraph line is the connection between the transmitter and the receiver; so if the echo matches the data sent, the program is certain that the data was on the telegraph lines. The program would receive a character as in steps 5 and 6 under receiving for echo checking. (The scanner is not released (clearing of Scanner flag) at this point unless action of step 7 is desired.) This may be followed by step 6.

6. The program may transmit another character. Note that the scanner counter is used, the scanner is released and the Receiver flag is cleared. To insure maximum transmission rates, the program must reload the transmitter buffer within the following times (after service has been requested).

WPM	Bit Code	Baud	Unit Code	Time
60.0	5	45.5	7.5	30.0 msec
66.6	5	50.0	7.5	27.5 msec
100.0	5	75.0	7.5	18.5 msec
100.0	8	110.0	11.0	17.0 msec

7. The program may desire to stop transmitting. The Receiver and Scanner flags must be cleared. This is accomplished by reading the character which clears the Receiver flag or by performing step 5 (echo check) and releasing the scanner (same as steps 5, 6, and 7 under receiving).

There are programming techniques which implement operator intervention and control such as: If the echo check feature is utilized in the program, the Teletype operator may signal the program while the computer is typing out. To accomplish this, the operator pushes a random key; this action causes an echo check in the computer program. The program then waits for instructions from the operator, and if none are received, resends the failed character.

The null/idle character in the 8-bit code may be distinguished from the character produced when the break key is pushed by examining the eighth (most significant) bit (1 equals null, 0 equals break if parity is not being generated). A large number of break characters would be interpreted by the program as an open line which may be detected by a series of blank characters in the 5-bit code.

The program must determine by a program switch which state the line is in (transmitting or receiving). To insure that the correct character (null, blank) is read when receiving on an open line, the program must examine the character within the times listed above in step 6.

The programming operations used to accomplish transmitting and receiving characters in a many station Teletype communication system involves knowing when stations require servicing, being able to determine the specific station that requires servicing, and transferring characters between the 630 DCS and the PDP-6.

The method of receiving a character is as follows:

1. The receiver turns on a flag at the completion of receiving a Teletype character.
2. A rotating priority type scanner using a counter is continually sampling the Receiver flags. When a flag is found on, the computer stops. The number in the counter is unique to the receiver requesting service. (The number may range from 00 to 77_8 depending upon the number of stations connected. For discussion purposes, it is suggested that all labeling, identification, and references to stations be in accordance with the following scheme:

There are up to eight stations in a group, numbered 0-7 and assigned the units position of a 2-digit octal number. There are up to eight groups in the scanner system, numbered 0-7 and assigned the eights position of the 2-digit octal number. Thus station 24 is station 4 in group 2.)

3. When the counter stops, a second flag (the Scanner flag) is turned on. It is this flag that notifies the computer that the scanner has found a receiver that requires service. Notification may be in the form of a program interrupt, program test by check status, and/or skip test of a flag.

4. The program requests that the number contained in the counter (station number) be placed into an input register (in-out register, accumulator register, core memory register). In some Type 633 models, the line number may be ORed to the input register at the program's option. (The program may release the scanner at this time. Doing so does not clear the receiver flag. After one complete revolution, the scanner finds the Receiver flag still on and again requests service for the station. Thus, the program may temporarily ignore a station request for service and permit the scanner to search for a station requiring higher priority service.) The program may reset the counter to zero. This in effect gives the lower numbered stations higher priority.
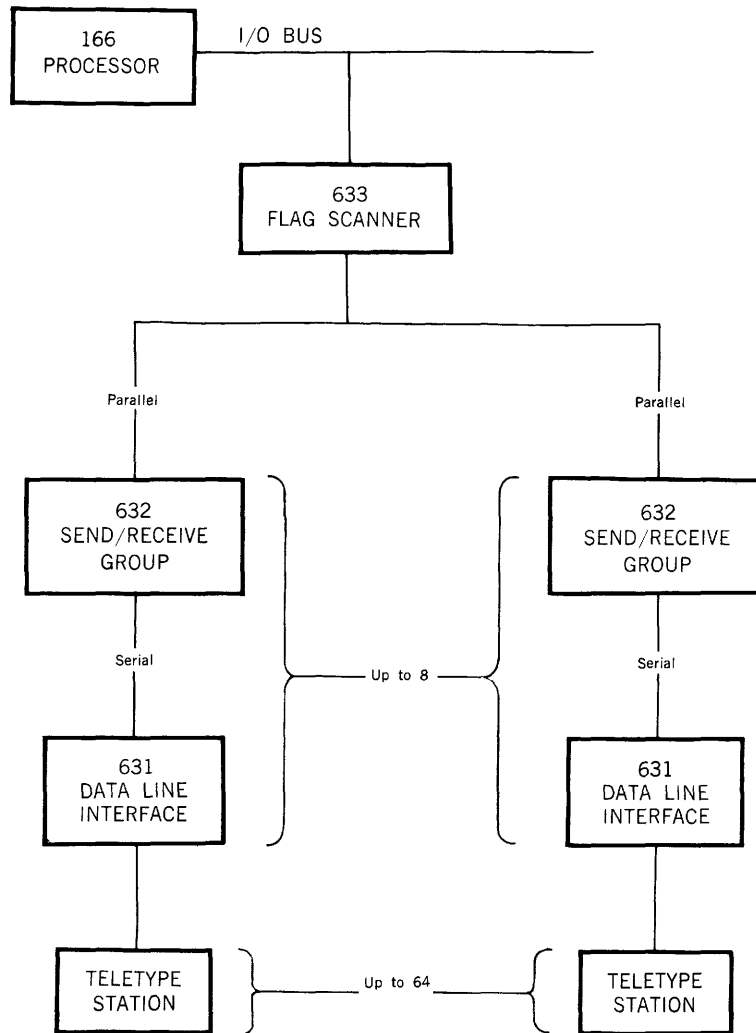
Figure 24 Data Communication System Type 630

5. The program then reads the character from the service requesting receiver into an input register (in-out register, accumulator register, core memory register). In some Type 633 models, the incoming character may be ORed to the input register at the program's option. If the "OR" option is used, that portion of the input register, where the character is received, must be zero prior to read-in. Transmitter garbling may otherwise occur.

6. Reading the character clears the Receiver flag.

7. The Scanner flag should be cleared at this time.

8. To insure that characters are not lost, when receiving at maximum data rates, the program must examine the character within the following times (after service has been requested).

WPM	Bit Code	Baud	Unit Code	Time
60.0	5	45.5	7.5	41.0 msec
66.6	5	50.0	7.5	37.5 msec
100.0	5	75.0	7.5	25.0 msec
100.0	8	110.0	11.0	21.5 msec

If delays greater than above are experienced, the character read in will be garbled; however, the next character received will be correct. The Receiver flag must be turned off (step A6) even if the program exceeds the above limits. Otherwise; the completion of the next character cannot be determined.

APPENDIX 1
INSTRUCTIONS

FULL WORD TRANSMISSION INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
MOVE		200	$C(E) => C(AC)$	
	I	201	$E => C(AC)$	
	M	202	$C(AC) => C(E)$	
	S	203	$C(E) => C(E)$	
MOVS		204	$C(E)_L => C(AC)_R$ $C(E)_R => C(AC)_L$	
	I	205	$0 => C(AC)_R$ $E => C(AC)_L$	
	M	206	$C(AC)_L => C(E)_R$ $C(AC)_R => C(E)_L$	
	S	207	$C(E)_L => C(E)_R$ $C(E)_R => C(E)_L$	
MOVN		210	$-C(E) => C(AC)$	
	I	211	$-E => C(AC)$	The negative is the twos complement
	M	212	$-C(AC) => C(E)$	
	S	213	$-C(E) => C(E)$	
MOVM		214	If $C(E) \geq 0$, then $C(E) => C(AC)$ If $C(E) < 0$, then $-C(E) => C(AC)$	May set AC OV flag
	I	215	$E => C(AC)$	
	M	216	If $C(AC) \geq 0$, then $C(AC) => C(E)$ If $C(AC) < 0$, then $-C(AC) => C(E)$	
	S	217	If $C(E) \geq 0$, then $C(E) => C(E)$ If $C(E) < 0$, then $-C(E) => C(E)$	

HALF WORD TRANSMISSION INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
HLL		500	$C(E)_L => C(AC)_L$	
	I	501	$0 => C(AC)_L$	
	M	502	$C(AC)_L => C(E)_L$	
	S	503	$C(E)_L => C(E)_L$	
HRR		540	$C(E)_R => C(AC)_R$	
	I	541	$E => C(AC)_R$	
	M	542	$C(AC)_R => C(E)_R$	
	S	543	$C(E)_R => C(E)_R$	

Mnemonic	Mode	Code	Effect	Comments
HRL		504	$C(E)_R => C(AC)_L$	
	I	505	$E => C(AC)_L$	
	M	506	$C(AC)_R => C(E)_L$	
	S	507	$C(E)_R => C(E)_L$	
HLR		544	$C(E)_L => C(AC)_R$	
	I	545	$0 => C(AC)_R$	
	M	546	$C(AC)_L => C(E)_R$	
	S	547	$C(E)_L => C(E)_R$	
HLLZ		510	$C(E)_L => C(AC)_L$ $0 => C(AC)_R$	
	I	511	$0 => C(AC)_L$ $0 => C(AC)_R$	
	M	512	$C(AC)_L => C(E)_L$ $0 => C(E)_R$	
	S	513	$C(E)_L => C(E)_L$ $0 => C(E)_R$	
HRRZ		550	$C(E)_R => C(AC)_R$ $0 => C(AC)_L$	
	I	551	$E => C(AC)_R$ $0 => C(AC)_L$	
	M	552	$C(AC)_R => C(E)_R$ $0 => C(E)_L$	
	S	553	$C(E)_R => C(E)_R$ $0 => C(E)_L$	
HLLO		520	$C(E)_L => C(AC)_L$ $777777_8 => C(AC)_R$	
	I	521	$0 => C(AC)_L$ $777777_8 => C(AC)_R$	
	M	522	$C(AC)_L => C(E)_L$ $777777_8 => C(E)_R$	
	S	523	$C(E)_L => C(E)_L$ $777777_8 => C(E)_R$	
HRRO		560	$C(E)_R => C(AC)_R$ $777777_8 => C(AC)_L$	
	I	561	$E => C(AC)_R$ $777777_8 => C(AC)_L$	
	M	562	$C(AC)_R => C(E)_R$ $777777_8 => C(E)_L$	
	S	563	$C(E)_R => C(E)_R$ $777777_8 => C(E)_L$	
HRLO		524	$C(E)_R => C(AC)_L$ $777777_8 => C(AC)_R$	
	I	525	$E => C(AC)_L$ $777777_8 => C(AC)_R$	
	M	526	$C(AC)_R => C(E)_L$ $777777_8 => C(E)_R$	
	S	527	$C(E)_R => C(E)_L$ $777777_8 => C(E)_R$	

HALF WORD TRANSMISSION INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
HLLE		530	$C(E)_L => C(AC)_L$ If $C(E)_0 = 0$, then $0 => C(AC)_R$ If $C(E)_0 = 1$, then $777777_8 => C(AC)_R$	
	I	531	$0 => C(AC)$	
	M	532	$C(AC)_L => C(E)_L$ If $C(AC)_0 = 0$, then $0 => C(E)_R$ If $C(AC)_0 = 1$, then $777777_8 => C(E)_R$	
	S	533	$C(E)_L => C(E)_L$ If $C(E)_0 = 0$, then $0 => C(E)_R$ If $C(E)_0 = 1$, then $777777_8 => C(E)_R$	
HRRE		570	$C(E)_R => C(AC)_R$ If $C(E)_{18} = 0$, then $0 => C(AC)_L$ If $C(E)_{18} = 1$, then $777777_8 => C(AC)_L$	
	I	571	$E => C(AC)_R$ If $Y_{18} = 0$, then $0 => C(AC)_L$ If $Y_{18} = 1$, then $777777_8 => C(AC)_L$	
	M	572	$C(AC)_R => C(E)_R$ If $C(AC)_{18} = 0$, then $0 => C(E)_L$ If $C(AC)_{18} = 1$, then $777777_8 => C(E)_L$	
	S	573	$C(E)_R => C(E)_R$ If $C(E)_{18} = 0$, then $0 => C(E)_L$ If $C(E)_{18} = 1$, then $777777_8 => C(E)_L$	
HRLZ		514	$C(E)_R => C(AC)_L$ $0 => C(AC)_R$	
	I	515	$E => C(AC)_L$ $0 => C(AC)_R$	
	M	516	$C(AC)_R => C(E)_L$ $0 => C(E)_R$	
	S	517	$C(E)_R => C(E)_L$ $0 => C(E)_R$	
HLRZ		554	$C(E)_L => C(AC)_R$ $0 => C(AC)_L$	
	I	555	$0 => C(AC)$	
	M	556	$C(AC)_L => C(E)_R$ $0 => C(E)_L$	
	S	557	$C(E)_L => C(E)_R$ $0 => C(E)_L$	

HALF WORD TRANSMISSION INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
HLRO		564	$C(E)_L => C(AC)_R$ $777777_8 => C(AC)_L$	
	I	565	$0 => C(AC)_R$ $777777_8 => C(AC)_L$	
	M	566	$C(AC)_L => C(E)_R$ $777777_8 => C(E)_L$	
	S	567	$C(E)_L => C(E)_R$ $777777_8 => C(E)_L$	
HRLE		534	$C(E)_R => C(AC)_L$ If $C(E)_{18} = 0$, then $0 => C(AC)_R$ If $C(E)_{18} = 1$, then $777777_8 => C(AC)_R$	
	I	535	$E => C(AC)_L$ If $Y_{18} = 0$, then $0 => C(AC)_R$ If $Y_{18} = 1$, then $777777_8 => C(AC)_R$	
	M	536	$C(AC)_R => C(E)_L$ If $C(AC)_{18} = 0$, then $0 => C(E)_R$ If $C(AC)_{18} = 1$, then $777777_8 => C(E)_R$	
	S	537	$C(E)_R => C(E)_L$ If $C(E)_{18} = 0$, then $0 => C(E)_R$ If $C(E)_{18} = 1$, then $777777_8 => C(E)_R$	
HLRE		574	$C(E)_L => C(AC)_R$ If $C(E)_0 = 0$, then $0 => C(AC)_L$ If $C(E)_0 = 1$, then $777777_8 => C(AC)_L$	
	I	575	$0 => C(AC)$	
	M	576	$C(AC)_L => C(E)_R$ If $C(AC)_0 = 0$, then $0 => C(E)_L$ If $C(AC)_0 = 1$, then $777777_8 => C(E)_L$	
	S	577	$C(E)_L => C(E)_R$ If $C(E)_0 = 0$, then $0 => C(E)_L$ If $C(E)_0 = 1$, then $777777_8 => C(E)_L$	

BYTE MANIPULATION INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments

IBP		133	If $C(E)_P \geq C(E)_S$, then
			$\quad C(E)_P - C(E)_S => C(E)_P$
			If $C(E)_P < C(E)_S$, then
			$\quad 36 - C(E)_S => C(E)_P$ and
			$\quad C(E)_R + 1 => C(E)_R$	

ILDB		134	$C(E)_P - C(E)_S => C(E)_P$ $\quad C(C(E)) => C(AC)_{JR}$	See the effective BIS flag on the increment instruction.
LDB		135	$C(C(E)) => C(AC)_{JR}$	
IDPB		136	$C(E)_P - C(E)_S => C(E)_P$ $\quad C(AC)_{JR} => C(C(E))$	See the effective BIS flag on the increment instruction.
DPB		137	$C(AC)_{JR} => C(C(E))$	

FIXED POINT ARITHMETIC INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
ADD		270	$C(E) + C(AC) => C(AC)$	
	I	271	$E + C(AC) => C(AC)$	May set the OV, CRY0, CRY1 flags
	M	272	$C(E) + C(AC) => C(E)$	
	B	273	$C(E) + C(AC) => C(E),$ $\quad C(AC)$	
SUB		274	$C(AC) - C(E) => C(AC)$	
	I	275	$C(AC) - E => C(AC)$	May set the OV, CRY0, CRY1 flags
	M	276	$C(AC) - C(E) => C(E)$	
	B	277	$C(AC) - C(E) => C(E),$ $\quad C(AC)$	
IMUL		220	$C(AC) \times C(E) => C(AC)$	
	I	221	$C(AC) \times E => C(AC)$	
	M	222	$C(AC) \times C(E) => C(E)$	May set OV flag
	B	223	$C(AC) \times C(E) => C(E),$	
IDIV		230	$C(AC)/C(E) => C(AC)$ $\quad$ Remainder $=> C(AC + 1)$	
	I	231	$C(AC)/E => C(AC)$ $\quad$ Remainder $=> C(AC + 1)$	
	M	232	$C(AC)/C(E) => C(E)$	May set OV flag
	B	233	$C(AC)/C(E) => C(E),$ $\quad C(AC),$ Rem $=> C(AC + 1)$	
MUL		224	$C(AC) \times C(E) => C(AC)$ $\quad$ Low order part to $\quad C(AC + 1)$	
	I	225	$C(AC) \times E => C(AC)$ $\quad$ Low order part to $\quad C(AC + 1)$	May set OV flag
	M	226	$C(AC) \times C(E) => C(E)$	
	B	227	$C(AC) \times C(E) => C(E),$ $\quad C(AC)$ $\quad$ Low order part to $\quad C(AC + 1)$	

FIXED POINT ARITHMETIC (continued)

Mnemonic	Mode	Code	Effect	Comments
DIV		234	C(AC), C(AC + 1)/C(E) => C(AC), Rem => C(AC + 1)	
	I	235	C(AC), C(AC + 1)/E => C(AC), Rem => C(AC + 1)	
	M	236	C(AC), C(AC + 1)/C(E) => C(E)	
	B	237	C(AC), C(AC + 1)/C(E) => C(E), C(AC), Rem => C(AC + 1)	May set OV flag

FLOATING POINT ARITHMETIC INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
FAD		140	C(AC) + C(E) => C(AC)	May set OV flag
	L	141	C(AC) + C(E) => C(AC), C(AC + 1)	Low order bits to AC + 1, Left adjusted with no
	M	142	C(AC) + C(E) => C(E)	exponent
	B	143	C(AC) + C(E) => C(E), C(AC)	
FSB		150	C(AC) − C(E) => C(AC)	May set OV flag
	L	151	C(AC) − C(E) => C(AC), C(AC + 1)	Low order bits to AC + 1, Left adjusted with no exponent
	M	152	C(AC) − C(E) => C(E)	
	B	153	C(AC) − C(E) => C(E), C(AC)	
FMP		160	C(AC) × C(E) => C(AC)	May set OV flag
	L	161	C(AC) × C(E) => C(AC), C(AC + 1)	Low order bits to AC + 1, Left adjusted with no
	M	162	C(AC) × C(E) => C(E)	exponent
	B	163	C(AC) × C(E) => C(E), C(AC)	
FDV		170	C(AC)/C(E) => C(AC)	May set OV flag
	L	171	C(AC)/C(E) => C(AC), C(AC + 1)	Remainder to AC + 1, Left adjusted with no
	M	172	C(AC)/C(E) => C(E)	exponent
	B	173	C(AC)/C(E) => C(E), C(AC)	
FADR		144	C(AC) + C(E) => C(AC)	Same as FAD except for
	L	145	C(AC) + C(E) => C(AC), C(AC + 1)	rounding Low order bits to AC + 1,
	M	146	C(AC) + C(E) => C(E)	Left adjusted with no
	B	147	C(AC) + C(E) => C(E), C(AC)	exponent, The results in AC are rounded
FSBR		154	C(AC) − C(E) => C(AC)	May set OV flag
	L	155	C(AC) − C(E) => C(AC)	Same as FAD except for
	M	156	C(AC) − C(E) => C(E)	rounding
	B	157	C(AC) − C(E) => C(E), C(AC)	Low order bits to AC + 1, Left adjusted with no exponent,

FLOATING POINT ARITHMETIC INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
FMPR		164	$C(AC) \times C(E) => C(AC)$	Same as FMP except for rounding.
	L	165	$C(AC) \times C(E) =>$ $C(AC), C(AC + 1)$	Low order bits to AC + 1, Left adjusted with no exponent.
	M	166	$C(AC) \times C(E) => C(E)$	
	B	167	$C(AC) \times C(E) => C(E),$ $C(AC)$	
FDVR		174	$C(AC)/C(E) => C(AC)$	May set OV flag
	L	175	$C(AC)/C(E) => C(AC),$ $C(AC + 1)$	Remainder to AC + 1, Left adjusted with no exponent.
	M	176	$C(AC)/C(E) => C(E)$	
	B	177	$C(AC)/C(E) => C(E),$ $C(AC)$	
FSC		132	$C(AC) \times 2^E => C(AC)$	E = Effective address. May set OV flag

BOOLEAN INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
CLEAR		400	$0 => C(AC)$	
	I	401	$0 => C(AC)$	
	M	402	$0 => C(E)$	
	B	403	$0 => C(E), C(AC)$	
AND		404	$C(E) \wedge C(AC) => C(AC)$	
	I	405	$E \wedge C(AC) => C(AC)$	
	M	406	$C(E) \wedge C(AC) => C(E)$	
	B	407	$C(E) \wedge C(AC) => C(E),$ $C(AC)$	
ANDCA		410	$C(E) \wedge C(\overline{AC}) => C(AC)$	
	I	411	$E \wedge C(\overline{AC}) => C(AC)$	
	M	412	$C(E) \wedge C(\overline{AC}) => C(E)$	
	B	413	$C(E) \wedge C(\overline{AC}) => C(E),$ $C(AC)$	
SETM		414	$C(E) => C(AC)$	
	I	415	$E => C(AC)$	
	M	416	$C(E) => C(E)$	
	B	417	$C(E) => C(E), C(AC)$	
ANDCM		420	$C(\overline{E}) \wedge C(AC) => C(AC)$	
	I	421	$\overline{E} \wedge C(AC) => C(AC)$	
	M	422	$C(\overline{E}) \wedge C(AC) => C(E)$	
	B	423	$C(\overline{E}) \wedge C(AC) => C(E),$ $C(AC)$	
SETA		424	$C(AC) => C(AC)$	
	I	425	$C(AC) => C(AC)$	
	M	426	$C(AC) => C(E)$	
	B	427	$C(AC) => C(E), C(AC)$	

BOOLEAN INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
XOR		430	$C(E) \veebar C(AC) => C(AC)$	
	I	431	$E \veebar C(AC) => C(AC)$	
	M	432	$C(E) \veebar C(AC) => C(E)$	
	B	433	$C(E) \veebar C(AC) => C(E),$ C(AC)	
IOR		434	$C(E) \vee C(AC) => C(AC)$	
	I	435	$E \vee C(AC) => C(AC)$	
	M	436	$C(E) \vee C(AC) => C(E)$	
	B	437	$C(E) \vee C(AC) => C(E),$ C(AC)	
ANDCB		440	$C(\overline{E}) \wedge C(\overline{AC}) => C(AC)$	
	I	441	$\overline{E} \wedge C(\overline{AC}) => C(AC)$	
	M	442	$C(\overline{E}) \wedge C(\overline{AC}) => C(E)$	
	B	443	$C(\overline{E}) \wedge C(\overline{AC}) => C(E),$ C(AC)	
EQV		444	$C(E) \equiv C(AC) => C(AC)$	
	I	445	$E \equiv C(AC) => C(AC)$	
	M	446	$C(E) \equiv C(AC) => C(E)$	
	B	447	$C(E) \equiv C(AC) => C(E),$ C(AC)	
SETCA		450	$C(\overline{AC}) => C(AC)$	
	I	451	$C(\overline{AC}) => C(AC)$	
	M	452	$C(\overline{AC}) => C(E)$	
	B	453	$C(\overline{AC}) => C(AC), C(E)$	
ORCA		454	$C(E) \vee C(\overline{AC}) => C(AC)$	
	I	455	$E \vee C(\overline{AC}) => C(AC)$	
	M	456	$C(E) \vee C(\overline{AC}) => C(E)$	
	B	457	$C(E) \vee C(\overline{AC}) => C(E),$ C(AC)	
SETCM		460	$C(\overline{E}) => C(AC)$	
	I	461	$\overline{E} => C(AC)$	
	M	462	$C(\overline{E}) => C(E)$	
	B	463	$C(\overline{E}) => C(E), C(AC)$	
ORCM		464	$C(\overline{E}) \vee C(AC) => C(AC)$	
	I	465	$\overline{E} \vee C(AC) => C(AC)$	
	M	466	$C(\overline{E}) \vee C(AC) => C(E)$	
	B	467	$C(\overline{E}) \vee C(AC) => C(E),$ C(AC)	
ORCB		470	$C(\overline{E}) \vee C(\overline{AC}) => C(AC)$	
	I	471	$\overline{E} \vee C(\overline{AC}) => C(AC)$	
	M	472	$C(\overline{E}) \vee C(\overline{AC}) => C(E)$	
	B	473	$C(\overline{E}) \vee C(\overline{AC}) => C(E),$ C(AC)	
SETO		474	set all bits of AC to 1	
	I	475	same as above	
	M	476	same to C(E)	
	B	477	same to C(E) and C(AC)	

SHIFT INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
ASH		240	$2^E \times C(AC) \Rightarrow C(AC)$	E = Effective address
ROT		241	$C(AC)_i \Rightarrow C(AC)_{i-E}$ If left, bits moving through $C(AC)_0 \Rightarrow C(AC)_{35}$ If right, bits moving through $C(AC)_{35} \Rightarrow C(AC)_0$	Sets AR OV flag if $C(AR)_0 \ \forall \ C(AR)_1 = 1$
LSH		242	$C(AC)_i \Rightarrow C(AC)_{i-E}$	
ASHC		244	$2^E \times C(CAC) \Rightarrow C(CAC)$	Sets AR OV flag if
ROTC		245	$C(CAC)_i \Rightarrow C(CAC)_{i-E}$ If left $C(AC)_0 \Rightarrow C(AC + 1)_{35}$ If right $C(AC + 1)_{35} \Rightarrow C(AC)_0$	$C(AR)_0 \ \forall \ C(AR)_1 = 1$
LSHC		246	$C(CAC)_i \Rightarrow C(CAC)_{i-E}$	

MEMORY AND ACCUMULATOR MODIFICATION AND TESTING INSTRUCTIONS

Instructions set the PC Change flag if skip or jump is executed.

Skip instructions test and/or modify the contents of the effective address.
If the condition specified by the mode control is met, the next instruction is skipped.

Jump instructions test and/or modify the accumulator. If conditions specified by the mode control are met, the next instruction is skipped.

Mnemonic	Moue	Code	Effect	Comments
JUMP	—	320	None	For an unconditional jump, this instruction is slower than JRST.
	L	321	If $C(AC) < 0$, then $E \Rightarrow C(PC)$	
	E	322	If $C(AC) = 0$, then $E \Rightarrow C(PC)$	
	G	327	If $C(AC) > 0$, then $E \Rightarrow C(PC)$	
	A	324	$E \Rightarrow C(PC)$	
	GE	325	If $C(AC) \geq 0$, then $E \Rightarrow C(PC)$	
	N	326	If $C(AC) \neq 0$, then $E \Rightarrow C(PC)$	
	LE	323	If $C(AC) \leq 0$, then $E \Rightarrow C(PC)$	
SKIP	—	330		
	L	331	If $C(E) < 0$, then skip	
	E	332	If $C(E) = 0$, then skip	
	G	337	If $C(E) > 0$, then skip	
	A	334	Always skip	
	GE	335	If $C(E) \geq 0$, then skip	

MEMORY AND ACCUMULATOR MODIFICATION AND TESTING
INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
	N	336	If C(E) $\neq$ 0, then skip	
	LE	333	If C(E) $\leq$ 0, then skip	
AOJ	—	340	C(AC) + 1 => C(AC)	AOJ instructions may set
	L	341	C(AC) + 1 => C(AC) If C(AC) < 0, then E => C(PC)	AC CRY0 and AC CRY1 flag
	E	342	C(AC) + 1 => C(AC) If C(AC) = 0, then E => C(PC)	
	G	347	C(AC) + 1 => C(AC) If C(AC) > 0, then E => C(PC)	
	A	344	C(AC) + 1 => C(AC) E => C(PC)	
	GE	345	C(AC) + 1 => C(AC) If C(AC) $\geq$ 0, then E => C(PC)	
	N	346	C(AC) + 1 => C(AC) If C(AC) $\neq$ 0, then E => C(PC)	
	LE	343	C(AC) + 1 => C(AC) If C(AC) $\leq$ 0, then E => C(PC)	
AOS	—	350	C(E) + 1 => C(E)	AOS instructions may set
	L	351	C(E) + 1 => C(E) If C(E) < 0, then skip	AC CRY0 and AC CRY1 flags
	E	352	C(E) + 1 => C(E) If C(E) = 0, then skip	
	G	357	C(E) + 1 => C(E) If C(E) > 0, then skip	
	A	354	C(E) + 1 => C(E)	
	GE	355	C(E) + 1 => C(E) If C(E) $\geq$ 0, then skip	
	N	356	C(E) + 1 => C(E) If C(E) $\neq$ 0, then skip	
	LE	353	C(E) + 1 => C(E) If C(E) $\leq$ 0, then skip	
SOJ	—	360	C(AC) − 1 => C(AC)	SOJ instructions may set
	L	361	C(AC) − 1 => C(AC) If C(AC) < 0, then E => C(PC)	AC CRY0 and AC CRY1 flags
	E	362	C(AC) − 1 => C(AC) If C(AC) = 0, then E => C(PC)	
	G	367	C(AC) − 1 => C(AC) If C(AC) > 0, then E => C(PC)	
	A	364	C(AC) − 1 => C(AC) E => C(PC)	

MEMORY AND ACCUMULATOR MODIFICATION AND TESTING
INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
	GE	365	$C(AC) - 1 \Rightarrow C(AC)$ If $C(AC) \geq 0$, then $\quad E \Rightarrow C(PC)$	
	N	366	$C(AC) - 1 \Rightarrow C(AC)$ If $C(AC) \neq 0$, then $\quad E \Rightarrow C(PC)$	
	LE	363	$C(AC) - 1 \Rightarrow C(AC)$ If $C(AC) \leq 0$, then $\quad E \Rightarrow C(PC)$	
SOS	—	370	$C(E) - 1 \Rightarrow C(E)$	SOS instructions may set
	L	371	$C(E) - 1 \Rightarrow C(E)$ If $C(E) < 0$; then skip	AC CRY0 and AC CRY1 flags
	E	372	$C(E) - 1 \Rightarrow C(E)$ If $C(E) = 0$, then skip	
	G	377	$C(E) - 1 \Rightarrow C(E)$ If $C(E) > 0$, then skip	
	A	374	$C(E) - 1 \Rightarrow C(E)$ and skip	
	GE	375	$C(E) - 1 \Rightarrow C(E)$ If $C(E) \geq 0$, then skip	
	N	376	$C(E) - 1 \Rightarrow C(E)$ If $C(E) \neq 0$, then skip	
	LE	373	$C(E) - 1 \Rightarrow C(E)$ If $C(E) \leq 0$, then skip	

ARITHMETIC COMPARE INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
CAM	—	310	Never skip. No effect	
	L	311	Skip if $C(AC) < C(E)$	
	E	312	Skip if $C(AC) = C(E)$	
	G	317	Skip if $C(AC) > C(E)$	
	A	314	Always skip	
	GE	315	Skip if $C(AC) \geq C(E)$	
	N	316	Skip if $C(AC) \neq C(E)$	
	LE	313	Skip if $C(AC) \leq C(E)$	
CAI	—	300	Never skip	
	L	301	Skip if $C(AC) < E$	
	E	302	Skip if $C(AC) = E$	
	G	307	Skip if $C(AC) > E$	
	A	304	Always skip	
	GE	305	Skip if $C(AC) \geq E$	
	N	306	Skip if $C(AC) \neq E$	
	LE	303	Skip if $C(AC) \leq E$	

LOGICAL COMPARE AND MODIFY INSTRUCTIONS

Instruction code format: 110 ww x yy z, where

> ww specifies whether to do nothing to the selected bits, to clear the selected bits, to set the selected bits to ones, or complement the selected bits.
>
> x specifies whether the bit selection is done by C(E) or E.
>
> yy specifies no skip, skip on zero, always skip, or skip on not zero.
>
> z specifies whether to use the data word directly or with the left and right halves exchanged.

Instructions set the PC Change flag if skip is executed.

Mnemonic	Mode	Code	Effect	Comments
TDN	—	610	None	
	E	612	If $C(E) \wedge C(AC) = 0$, then skip.	
	A	614	Always skips.	
	N	616	If $C(E) \wedge C(AC) \neq 0$, then skip.	
TSN	—	611	None	
	E	613	If $C(E)_S \wedge C(AC) = 0$, then skip.	
	A	615	Always skips.	
	N	617	If $C(E)_S \wedge C(AC) \neq 0$, then skip.	
TRN	—	600	None	
	E	602	If $E \wedge C(AC)_R = 0$, then skip.	
	A	604	Always skips.	
	N	606	If $E \wedge C(AC)_R \neq 0$, then skip.	
TLN	—	601	None	
	E	603	If $E \wedge C(AC)_L = 0$, then skip.	
	A	605	Always skips.	
	N	607	If $E \wedge C(AC)_L \neq 0$, then skip.	
TDZ	—	630	Clear selected bits. Do not skip.	
	E	632	If $C(E) \wedge C(AC) = 0$, then clear and skip; otherwise, clear and don't skip.	
	A	634	Clear selected bits and skip.	
	N	636	If $C(E) \wedge C(AC) \neq 0$, then clear and skip; otherwise, clear and don't skip.	
TSZ	—	631	Clear and don't skip.	
	E	633	If $C(E)_S \wedge C(AC) = 0$, then clear and skip; otherwise, clear and don't skip.	
	A	635	Clear and skip.	
	N	637	If $C(E)_S \wedge C(AC) \neq 0$, then clear and skip; otherwise, don't skip.	
TRZ	—	620	Clear and don't skip.	
	E	622	If $E \wedge C(AC)_R = 0$, then clear and skip; otherwise, clear and don't skip.	

LOGICAL COMPARE AND MODIFY INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
	A	624	Clear and skip.	
	N	626	If $E \wedge C(AC)_R \neq 0$, then clear and skip; otherwise, clear and don't skip.	
TLZ	—	621	Clear and don't skip.	
	E	623	If $E \wedge C(AC)_L = 0$, then clear and skip; otherwise, clear and don't skip.	
	A	625	Clear and skip.	
	N	627	If $E \wedge C(AC)_L \neq 0$, then clear and skip; otherwise, clear and don't skip.	
TDO	—	670	Set and don't skip.	
	E	672	If $C(E) \wedge C(AC) = 0$, then set and skip; otherwise, set and don't skip.	
	A	674	Set and skip.	
	N	676	If $C(E) \wedge C(AC) \neq 0$, then set and skip; otherwise, set and don't skip.	
TSO	—	671	Set and don't skip.	
	E	673	If $C(E)_S \wedge C(AC) = 0$, then set and skip; otherwise, set and don't skip.	
	A	675	Set and skip.	
	N	677	If $C(E)_S \wedge C(AC) \neq 0$, then set and skip; otherwise, set and don't skip.	
TRO	—	660	Set and don't skip.	
	E	662	If $E \wedge C(AC)_R = 0$, then set and skip; otherwise, set and don't skip.	
	A	664	Set and skip.	
	N	666	If $E \wedge C(AC)_R \neq 0$, then set and skip; otherwise, set and don't skip.	
TLO	—	661	Set and don't skip.	
	E	663	If $E \wedge C(AC)_{L'} = 0$, then set and skip; otherwise, set and don't skip.	
	A	665	Set and skip.	
	N	667	If $E \wedge C(AC)_L \neq 0$, then set and skip; otherwise, set and don't skip.	
TDC	—	650	Complement and don't skip.	
	E	652	If $C(E) \wedge C(AC) = 0$, then complement and skip; otherwise, complement and don't skip.	
	A	654	Complement and skip.	

Mnemonic	Mode	Code	Effect	Comments
	N	656	If $C(E) \wedge C(AC) \neq 0$, then complement and skip; otherwise, complement and don't skip.	
TSC	—	651	Complement and don't skip.	
	E	653	If $C(E)_S \wedge C(AC) = 0$, then complement and skip; otherwise, complement and don't skip.	
	A	655	Complement and skip.	
	N	657	If $C(E)_S \wedge C(AC) \neq 0$, then complement and skip; otherwise, complement and don't skip.	
TRC	—	640	Complement and don't skip.	
	E	642	If $E \wedge C(AC)_R = 0$, then complement and skip; otherwise, complement and don't skip.	
	A	644	Complement and skip.	
	N	646	If $E \wedge C(AC)_R \neq 0$, then complement and skip; otherwise, complement and don't skip.	
TLC	—	641	Complement and don't skip.	
	E	643	If $E \wedge C(AC)_L = 0$, then complement and skip; otherwise, complement and don't skip.	
	A	645	Complement and skip.	
	N	647	If $E \wedge C(AC)_L \neq 0$, then complement and skip; otherwise, complement and don't skip.	

JUMP AND PUSH DOWN INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
PUSHJ		260	$PC => C(C(AC)_R + 1)$ $C(AC) + 1000001_8 => C(AC)$ $E => PC$	When stored, program counter contains the location of the instruction $+1$.
PUSH		261	$C(E) => C(C(AC)_R + 1)$ $C(AC) + 1000001_8 => C(AC)$	For all push and pop instructions a carry from AR_0 sets the PDL OV flag (see CONO APR, and CONI APR).
POP		262	$C(C(AC)_R - 1) => C(E)$ $C(AC) - 1000001_8 => C(AC)$	
POPJ		263	$C(C(AC)_R - 1) => PC$ $C(AC) - 1000001_8 => C(AC)$	
JSR		264	OV, CRY0, CRY1, BIS, PC Change flag and $PC => C(E)$ $0 => BIS$ flag $E + 1 => C(PC)$	Flags stored in order shown in bits 0-5. PC stored in bits 13-35. When stored, program counter contains the location of the instruction $+1$.

JUMP AND PUSH DOWN INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
JSP		265	PC => C(AC) E => C(PC)	When stored, program counter contains the location of the instruction +1.
JSA		266	C(AC) => C(E) E => C(AC)$_L$ PC => C(AC)$_R$ E + 1 => C(PC)	When stored, program counter contains the location of the instruction +1.
JRA		267	C(C(AC)$_{0-17}$) => C(AC) E => C(PC)	

I/O INSTRUCTIONS

Mnemonic	Mode	Code yy y	Effect	Comments

BLKI		0 0	C(E) + 1000001 => C(E)
			If C(E)$_L$ ≠ 0, then
			a) if in sequence break mode, then
			(IOT Data) => C(C(E)$_R$)
			and dismiss break.
			b) if not in sequence break mode, then
			(IOT Data) => C(C(E)$_R$)
			and skip.
			If C(E)$_L$ = 0, then
			(IOT Data) => C(C(E)$_R$)
			and execute next instruction.	

DATAI		0 4	(IOT Data) => C(E) when ready.	If the device number is 000 000 0, then the instruction is assigned to the processor and reads the C(data switches).

BLKO		1 0	C(E) + 1000001 => C(E)
			If C(E)$_L$ ≠ 0, then
			a) if in sequence break mode, then
			C(C(E)$_R$) => (IOT Data)
			and dismiss break.
			b) if not in sequence break mode, then
			C(C(E)$_R$) => (IOT Data)
			and skip.
			If C(E)$_L$ = 0, then
			C(C(E)$_R$) => (IOT Data)
			and execute next instruction.	

I/O INSTRUCTIONS (continued)

Mnemonic	Mode	Code yy y	Effect	Comments
DATAO		1 4	C(E) => (IOT Data) when ready.	
CONO		2 0	E => device control register	If the device number is 000 000 0, the instruction is assigned to the processor. If the device number is 000 000 1, the instruction is assigned to the processor-PI system.
CONI		2 4	Device control register => C(E) when ready	If the device number is 000 000 0, then the instruction is assigned to the processor. If the device number is 000 000 1, then the instruction is assigned to the processor-PI system.
CONSZ		3 0	If device control register $\wedge$ E = 0, then skip.	The comments for CONI apply to this instruction.
CONSO		3 4	If device control register $\wedge$ E $\neq$ 0, then skip.	The comments for CONI apply to this instruction.

MISCELLANEOUS INSTRUCTIONS

Mnemonic	Mode	Code	Effect	Comments
EXCH		250	C(E) <=> C(AC)	
BLT		251	BLOCK (C(AC)$_L$ => BLOCK (C(AC)$_R$)	
AOBJP		252	C(AC) + 1000001 => C(AC) If C(AC) $\geq$ 0, then jump	This instruction will set the PC Change flag if the jump is executed. It may set the accumulator OV, CRY0, and CRY1 flags.
AOBJN		253	C(AC) + 1000001 => C(AC) If C(AC) < 0, then jump	This instruction will set the PC Change flag if the jump is executed. It may set the accumulator OV, CRY0, and CRY1 flags.

MISCELLANEOUS INSTRUCTIONS (continued)

Mnemonic	Mode	Code	Effect	Comments
JRST		254	Jump and: if bit 9 = 1, then reset or enable the current interrupt priority channel if bit 10 = 1, then halt if bit 11 = 1, then reset OV, CRY0, and CRY1 flags and PC Change flag. bit 12 is unused.	Will set PC Change flag. Bits 9–12 are set by giving an AC number.
JFCL		255	If flags are set, then jump and clear flags.	Flags are selected as follows: Bit 9 = OV Bit 10 = CRY0 Bit 11 = CRY1 Bit 12 = PC Change
XCT		256	Execute instruction at E	

APPENDIX 2
CODES

NUMERICAL

Octal Code	Mnemonic Code	Mode	Page	Octal Code	Mnemonic Code	Mode	Page
132	FSC		31, 91	211	MOVN	I	26, 85
133	IBP		28, 89	212	MOVN	M	26, 85
134	ILDB I		28, 89	213	MOVN	S	26, 85
135	IDPB I		28, 89	214	MOVM		26, 85
136	IDPB I		28, 89	215	MOVM	I	26, 85
137	IDPB I		28, 89	216	MOVM	M	26, 85
140	FAD	L	31, 90	217	MOVM	S	26, 85
141	FAD	M	31, 90	220	IMUL		30, 89
142	FAD	B	31, 90	221	IMUL	I	30, 89
143	FAD		31, 90	222	IMUL	M	30, 89
144	FADR	L	31, 90	223	IMUL	B	30, 89
145	FADR	M	31, 90	224	MUL		29, 89
146	FADR	B	31, 90	225	MUL	I	30, 89
147	FADR		31, 90	226	MUL	M	30, 89
150	FSB	L	31, 90	227	MUL	B	30, 89
151	FSB	M	31, 90	230	IDIV		30, 89
152	FSB	B	31, 90	231	IDIV	I	30, 89
153	FSB		31, 90	232	IDIV	M	30, 89
154	FSBR	L	31, 90	233	IDIV	B	30, 89
155	FSBR	M	31, 90	234	DIV		30, 90
156	FSBR	B	31, 90	235	DIV	I	30. 90
157	FSBR		31, 90	236	DIV	M	30. 90
160	FMP	L	31, 90	237	DIV	B	30, 90
161	FMP	M	31, 90	240	ASH		33, 93
162	FMP	B	31, 90	241	ROT		33, 93
163	FMP		31, 90	242	LSH		33, 93
164	FMPR	L	31, 91	243			
165	FMPR	M	31, 91	244	ASHC		33, 93
166	FMPR	B	31, 91	245	ROTC		33, 93
167	FMPR		31, 91	246	LSHC		33, 93
170	FDV	L	31, 90	247			
171	FDV	M	31, 90	250	EXCH		28, 100
172	FDV	B	31, 90	251	BLT		28, 100
173	FDV		31, 90	252	AOBJP		36, 100
174	FDVR	L	31, 91	253	AOBJN		36, 100
175	FDVR	M	31, 91	254	JRST		36, 100
176	FDVR	B	31, 91	255	JFCL		37, 101
177	FDVR		31, 91	256	XCT		37, 101
200	MOVE	I	26, 85	257			
201	MOVE	M	26, 85	260	PUSHJ		37, 98
202	MOVE	S	26, 85	261	PUSH		37, 98
203	MOVE		26, 85	262	POP		37, 98
204	MOVS	I	26, 85	263	POPJ		37, 98
205	MOVS	M	26, 85	264	JSR		36, 99
206	MOVS	S	26, 85	265	JSP		36, 99
207	MOVS		26, 85	266	JSA		36, 99
210	MOVN		26, 85	267	JRA		36, 99

Octal Code	Mnemonic Code	Mode	Page	Octal Code	Mnemonic Code	Mode	Page
270	ADD		29, 90	354	AOS	A	33, 94
271	ADD	I	30, 90	355	AOS	GE	33, 94
272	ADD	M	30, 90	356	AOS	N	33, 94
273	ADD	B	30, 90	357	AOS	G	33, 94
274	SUB		29, 90	360	SOJ	—	33, 94
275	SUB	I	30, 90	361	SOJ	L	33, 94
276	SUB	M	30, 90	362	SOJ	E	33, 94
277	SUB	B	30, 90	363	SOJ	LE	33, 95
300	CAI	—	34, 95	364	SOJ	A	33, 94
301	CAI	L	34, 95	365	SOJ	GE	33, 95
302	CAI	E	34, 95	366	SOJ	N	33, 95
303	CAI	LE	34, 95	367	SOJ	G	33, 94
304	CAI	A	34, 95	370	SOS	—	33, 95
305	CAI	GE	34, 95	371	SOS	L	33, 95
306	CAI	N	34, 95	372	SOS	E	33, 95
307	CAI	G	34, 95	373	SOS	LE	33, 95
310	CAM	—	34, 95	374	SOS	A	33, 95
311	CAM	L	34, 95	375	SOS	GE	33, 95
312	CAM	E	34, 95	376	SOS	N	33, 95
313	CAM	LE	34, 95	377	SOS	G	33, 95
314	CAM	A	34, 95	400	SETZ		32, 91
315	CAM	GE	34, 95	401	SETZ	I	32, 91
316	CAM	N	34, 95	402	SETZ	M	32, 91
317	CAM	G	34, 95	403	SETZ	B	32, 91
320	JUMP	—	33, 93	404	AND		32, 91
321	JUMP	L	33, 93	405	AND	I	32, 91
322	JUMP	E	33, 93	406	AND	M	32, 91
323	JUMP	LE	33, 93	407	AND	B	32, 91
324	JUMP	A	33, 93	410	ANDCA		32, 91
325	JUMP	GE	33, 93	411	ANDCA	I	32, 91
326	JUMP	N	33, 93	412	ANDCA	M	32, 91
327	JUMP	G	33, 93	413	ANDCA	B	32, 91
330	SKIP	—	33, 93	414	SETM		32, 91
331	SKIP	L	33, 93	415	SETM	I	32, 91
332	SKIP	E	33, 93	416	SETM	M	32, 91
333	SKIP	LE	33, 94	417	SETM	B	32, 91
334	SKIP	A	33, 93	420	ANDCM		32, 91
335	SKIP	GE	33, 93	421	ANDCM	I	32, 91
336	SKIP	N	33, 94	422	ANDCM	M	32, 91
337	SKIP	G	33, 93	423	ANDCM	B	32, 91
340	AOJ	—	33, 94	424	SETA		32, 91
341	AOJ	L	33, 94	425	SETA	I	32, 91
342	AOJ	E	33, 94	426	SETA	M	32, 91
343	AOJ	LE	33, 94	427	SETA	B	32, 91
344	AOJ	A	33, 94	430	OR		32, 92
345	AOJ	GE	33, 94	431	OR	I	32, 92
346	AOJ	N	33, 94	432	OR	M	32, 92
347	AOJ	G	33, 94	433	OR	B	32, 92
350	AOS	—	33, 94	434	IOR		32, 92
351	AOS	L	33, 94	435	IOR	I	32, 92
352	AOS	E	33, 94	436	IOR	M	32, 92
353	AOS	LE	33, 94	437	IOR	B	32, 92

Octal Code	Mnemonic Code	Mode	Page	Octal Code	Mnemonic Code	Mode	Page
440	ANDCB		32, 92	524	HRLO	I	27, 86
441	ANDCB	I	32, 92	525	HRLO	I	27, 86
442	ANDCB	M	32, 92	526	HRLO	M	27, 86
443	ANDCB	B	32, 92	527	HRLO	S	27, 86
444	EQV		32, 92	530	HLLE		27, 87
445	EQV	I	32, 92	531	HLLE	I	27, 87
446	EQV	M	32, 92	532	HLLE	M	27, 87
447	EQV	B	32, 92	533	HLLE	S	27, 87
450	SETCA		32, 92	534	HRLE		27, 88
451	SETCA	I	32, 92	535	HRLE	I	27, 88
452	SETCA	M	32, 92	536	HRLE	M	27, 88
453	SETCA	B	32, 92	537	HRLE	S	27, 88
454	ORCA		32, 92	540	HRR		26, 85
455	ORCA	I	32, 92	541	HRR	I	26, 85
456	ORCA	M	32, 92	542	HRR	M	26, 85
457	ORCA	B	32, 92	543	HRR	S	26, 85
460	SETCM		32, 92	544	HLR		26, 86
461	SETCM	I	32, 92	545	HLR	I	26, 86
462	SETCM	M	32, 92	546	HLR	M	26, 86
463	SETCM	B	32, 92	547	HLR	S	26, 86
464	ORCM		32, 92	550	HRRZ		26, 86
465	ORCM	I	32, 92	551	HRRZ	I	26, 86
466	ORCM	M	32, 92	552	HRRZ	M	26, 86
467	ORCM	B	32, 92	553	HRRZ	S	26, 86
470	ORCB		32, 92	554	HLRZ		26, 87
471	ORCB	I	32, 92	555	HLRZ	I	26, 87
472	ORCB	M	32, 92	556	HLRZ	M	26, 87
473	ORCB	B	32, 92	557	HLRZ	S	26, 87
474	SETO		32, 92	560	HRRO		27, 86
475	SETO	I	32, 92	561	HRRO	I	27, 86
476	SETO	M	32, 92	562	HRRO	M	27, 86
477	SETO	B	32, 92	563	HRRO	S	27, 86
500	HLL		26, 85	564	HLRO		27, 88
501	HLL	I	26, 85	565	HLRO	I	27, 88
502	HLL	M	26, 85	566	HLRO	M	27, 88
503	HLL	S	26, 85	567	HLRO	S	27, 88
504	HRL		26, 86	570	HRRE		27, 87
505	HRL	I	26, 86	571	HRRE	I	27, 87
506	HRL	M	26, 86	572	HRRE	M	27, 87
507	HRL	S	26, 86	573	HRRE	S	27, 87
510	HLLZ		26, 86	574	HLRE		27, 88
511	HLLZ	I	26, 86	575	HLRE	I	27, 88
512	HLLZ	M	26, 86	576	HLRE	M	27, 88
513	HLLZ	S	26, 86	577	HLRE	S	27, 88
514	HRLZ		26, 87	600	TRN	—	35, 96
515	HRLZ	I	26, 87	601	TLN	—	35, 96
516	HRLZ	M	26, 87	602	TRN	E	35, 96
517	HRLZ	S	26, 87	603	TLN	E	35, 96
520	HLLO		27, 86	604	TRN	A	35, 96
521	HLLO	I	27, 86	605	TLN	A	35, 96
522	HLLO	M	27, 86	606	TRN	N	35, 96
523	HLLO	S	27, 86	607	TLN	N	35, 96

Octal Code	Mnemonic Code	Mode	Page	Octal Code	Mnemonic Code	Mode	Page
610	TDN	—	35, 96	650	TDC	—	35, 97
611	TSN	—	35, 96	651	TSC	—	35, 98
612	TDN	E	35, 96	652	TDC	E	35, 97
613	TSN	E	35, 96	653	TSC	E	35, 98
614	TDN	A	35, 96	654	TDC	A	35, 97
615	TSN	A	35, 96	655	TSC	A	35, 98
616	TDN	N	35, 96	656	TDC	N	35, 97
617	TSN	N	35, 96	657	TSC	N	35, 98
620	TRZ	—	35, 96	660	TRO	—	35, 97
621	TLZ	—	35, 97	661	TLO	—	35, 97
622	TRZ	E	35, 96	662	TRO	E	35, 97
623	TLZ	E	35, 97	663	TLO	E	35, 97
624	TRZ	A	35, 96	664	TRO	A	35, 97
625	TLZ	A	35, 97	665	TLO	A	35, 97
626	TRZ	N	35, 96	666	TRO	N	35, 97
627	TLZ	N	35, 97	667	TLO	N	35, 97
630	TDZ	—	35, 96	670	TDO	—	35, 97
631	TSZ	—	35, 96	671	TSO	—	35, 97
632	TDZ	E	35, 96	672	TDO	E	35, 97
633	TSZ	E	35, 96	673	TSO	E	35, 97
634	TDZ	A	35, 96	674	TDO	A	35, 97
635	TSZ	A	35, 96	675	TSO	A	35, 97
636	TDZ	N	35, 96	676	TDO	N	35, 97
637	TSZ	N	35, 96	677	TSO	N	35, 97
640	TRC	—	35, 98	7 00	BLKI		38, 99
641	TLC	—	35, 98	7 04	DATAI		38, 99
642	TRC	E	35, 98	7 10	BLKO		38, 99
643	TLC	E	35, 98	7 14	DATAO		38, 100
644	TRC	A	35, 98	7 20	CONO		38, 100
645	TLC	A	35, 98	7 24	CONI		38, 100
646	TRC	N	35, 98	7 30	CONSZ		38, 100
647	TLC	N	35, 98	7 34	CONSO		38, 100

ALPHABETICAL

Mnemonic Code	Octal Code	Page		Mnemonic Code	Octal Code	Page
ADD	270-273	29, 89		IDIV	230-233	30, 89
AND	404-407	32, 91		IDPB	136	28, 89
ANDCA	410-413	32, 91		ILDB	134	30, 89
ANDCB	440-443	32, 92		IMUL	220-223	28, 89
ANDCM	420-423	32, 91		IOR	434-437	32, 92
AOBJN	253	36, 100		JFCL	255	37, 101
AOBJP	252	36, 100		JRA	267	36, 99
AOJ	340-347	33, 94		JRST	254	36, 100
AOS	350-357	33, 94		JSA	266	36, 99
ASH	240	33, 93		JSP	265	36, 99
ASHC	244	33, 93		JSR	264	36, 99
BLKI	700	38, 99		JUMP	320-327	33, 93
BLKO	710	38, 99		LDB	135	28, 89
BLT	251	28, 100		LSH	242	33, 93
CAI	300-307	34, 95		LSHC	246	33, 93
CAM	310-317	34, 95		MOVE	200-203	26, 85
CONI	724	38, 100		MOVM	214-217	26, 85
CONO	720	38, 100		MOVN	210-213	26, 85
CONSO	734	38, 100		MOVS	204-207	26, 85
CONSZ	730	38, 100		MUL	224-227	29, 89
DATAI	704	38, 99		OR	430-433	32, 92
DATAO	714	38, 100		ORCA	454-457	32, 92
DIV	234-237	30, 90		ORCB	470-473	32, 92
DPB	137	28, 89		ORCM	464-467	32, 92
EQV	444-447	32, 92		POP	262	37, 98
EXCH	250	28, 100		POPJ	263	37, 98
FAD	140-143	31, 90		PUSH	261	37, 98
FADR	144-147	31, 90		PUSHJ	260	37, 98
FDV	170-173	31, 90		ROT	241	33, 93
FDVR	174-177	31, 91		ROTC	245	33, 93
FMP	160-163	31, 90		SETA	424-427	32, 91
FMPR	164-167	31, 91		SETCA	450-453	32, 92
FSB	150-153	31, 90		SETCM	460-463	32, 92
FSBR	154-157	31, 90		SETM	414-417	32, 91
FSC	132	31, 91		SETO	474-477	32, 92
HLL	500-503	26, 85		SETZ	400-403	32, 91
HLLE	530-533	27, 87		SKIP	330-337	33, 93
HLLO	520-523	27, 86		SOJ	360-367	33, 94
HLLZ	510-513	26, 86		SOS	370-377	33, 95
HLR	544-547	26, 86		SUB	274-277	29, 89
HLRE	574-577	27, 88		TDC	650, 652, 654, 656	35, 97
HLRO	564-567	27, 88		TDN	610, 612, 614, 616	35, 96
HLRZ	554-557	26, 87				
HRL	504-507	26, 86		TDO	670, 672, 674, 676	35, 97
HRLE	534-537	27, 88				
HRLO	524-527	27, 88		TDZ	630, 632, 634, 636	35, 96
HRLZ	514-517	26, 87				
HRR	540-543	26, 85		TLC	641, 643, 645, 647	35, 98
HRRE	570-573	27, 87				
HRRO	560-563	27, 86		TLN	601, 603, 605, 607	35, 96
HRRZ	550-553	26, 86				
IBP	133	28, 89				

ALPHABETICAL (continued)

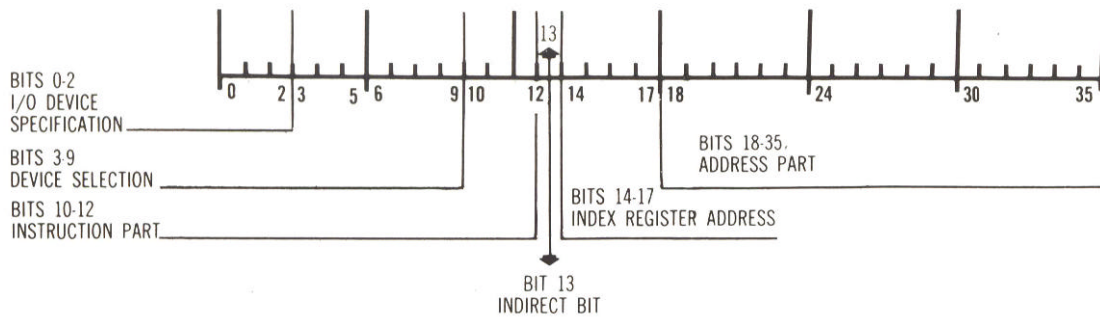| Mnemonic Code	Octal Code	Page	Mnemonic Code	Octal Code	Page
TLO	661, 663, 665, 667	35, 97	TSC	651, 653, 655, 657	35, 98
TLZ	621, 623, 625, 627	35, 97	TSN	611, 613, 615, 617	35, 96
TRC	640, 642, 644, 646	35, 98	TSO	671, 673, 675, 677	35, 97
TRN	600, 602, 604, 606	35, 96	TSZ	631, 633, 635, 637	35, 96
TRO	660, 662, 664, 666	35, 97	XCT	256	37, 101
TRZ	620, 622, 624, 626	35, 96			

APPENDIX 3
GLOSSARY OF ABBREVIATIONS

A — Always
AC — Accumulator (in Memory)
AR — Accumulator Register (in Arithmetic Processor)
B — Both
C — Contents of
CRY0 — Carry 0 flag
CRY1 — Carry 1 flag
E — Effective address
E — Equal to (if used as a mode indicator)
G — Greater than
GE — Greater than or equal to
I — Immediate
i — Any given bit (used as subscript)
JR — Justified right (starting in bit 35)
L — Less than (Skip Mode)
L — Low order part and /or remainder (Transmission Mode)
L — Left half (when used as subscript)
LE — Less than or equal to
M — Memory
N — Not equal to
OV — Overflow flag
P — (used as a subscript only) The P field of the pointer word
PC — Program counter
R — Right half
S — Self
S — Swapped Left and right halves exchanged
S — (used as a subscript only in Byte Manipulation) The S field of the pointer word.
— — Never skip
A — Logical complement
A=>B — A replaces B
= — Equivalent
∀ — Exclusive OR
V — Inclusive OR
∧ — AND

INPUT/OUTPUT SUMMARY CHART

INPUT-OUTPUT INSTRUCTION WORD FORMAT

BITS 0-2 I/O DEVICE SPECIFICATION
BITS 3-9 DEVICE SELECTION
BITS 10-12 INSTRUCTION PART
BIT 13 INDIRECT BIT
BITS 14-17 INDEX REGISTER ADDRESS
BITS 18-35 ADDRESS PART

(Bit scale: 0 2 3 5 6 9 10 12 13 14 17 18 24 30 35)

DEVICE / SPEED	OP CODE	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	INPUT BITS / OUTPUT BITS	INPUT INSTRUCTION EFFECT (DATAI, BLKI) / OUTPUT INSTRUCTION EFFECT (DATAO, BLKO)
166 CENTRAL PROCESSOR (CPA)	000	PDL OV FLAG	I/O RESET			MEMORY PROTECTION FLAG	NON-EX MEM FLAG	CLOCK DISABLE	CLOCK ENABLE	CLOCK FLAG OFF	PC CHG DISABLE	PC CHG ENABLE	PC CHG FLAG	AR OV DISABLE	AR OV ENABLE	AR OV FLAG	BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			36	
166 PRIORITY INTERRUPT SYSTEM (PRS)	004					PI CLEAR SYSTEM	PI ACTIVATE SELECTED	PI TURN SELECTED ON	PI TURN SELECTED OFF	PI TURN OFF	PI TURN ON	BITS 29-35 SEVEN PRIORITY INTERRUPT LINES (SELECT CHANNEL 1-7)									
761 PERFORATED TAPE PUNCH (PTP) — 63.3 CPS 15.8 MS/CHAR	100												ALPHA/BINARY 0=α 1=β	BUSY FLAG	DONE FLAG †	BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT				6/8 30-35	CLEAR DONE FLAG SET BUSY FLAG LOAD PUNCH BUFFER
760 PERFORATED TAPE READER (PTR) — 400 CPS 2.5 MS/CHAR 15 MS/WORD	104											POWER OFF/ON 0=OFF 1=ON	ALPHA/BINARY 0=α 1=β	BUSY FLAG	DONE FLAG †	BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT				8/36	CONTENTS OF READ BUFFER TO PROCESSOR SET BUSY FLAG CLEAR DONE FLAG (GET NEXT CHAR/WORD)
460 CARD PUNCH (CP) — 100 CPM	110									CARD ERROR	HOPPER EMPTY	STACKER FULL	NOT READY	EMPTY BIT	BUSY FLAG	DONE FLAG †	BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			36	LOAD BUFFER, CLEAR CP DONE FLAG
461 CARD READER (CR) — 200/800 CPM 2MS PER COL.	114							NOT READY	ERROR	END OF FILE	END OF CARD	DATA MISSED	ALL FLAG	HOLLERITH OR BINARY †	BUSY FLAG	DONE FLAG	BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			6·12 BITS PER COLUMN PACKED IN 36	INPUT DATA, CLEAR CR COL DONE FLAG
626 CONSOLE TELEPRINTER (TTY) — 10 CPS	120							TTI BUSY OFF	TTI FLAG OFF	TTO BUSY OFF	TTO FLAG OFF	TTI BUSY ON	TTI FLAG ON †	TTO BUSY ON	TTO FLAG ON †		BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			8 / 8	INPUT CHAR CLEAR BUSY FLAG CLEAR INPUT FLAG / OUTPUT CHAR SET BUSY FLAG CLEAR OUTPUT FLAG
646, 680 LINE PRINTERS (LPT)	124									LINE OVERFLOW ERROR †	ERROR †	BUSY FLAG	DONE FLAG †	BITS 30-32 PRIORITY INTERRUPT ERROR FOR FLAG			BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			36	
136 DATA CONTROL (DC)	200			BITS 20-22 NUMBER OF CHARACTERS ASSEMBLED —1			DATA MISSED	BUFFER ACCUMULATOR MOVE	DATA ACCUMULATOR REQUEST	DATA BUFFER REQUEST	IN/OUT	BITS 28-29 PACKING MODE (6/36 12/18 BIT MODE)		BITS 30-32 DEVICE NUMBER			BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			36 / 36	CONTENTS OF DATA BUFFER B TO PROCESSOR CLEAR DATA BUFFER REQUEST FLAG / PROC DATA TO DATA BUF B; CLEAR DB RU FLAG PROC DATA TO DATA BUFFER B CLEAR DATA BUF REQ FLAG
346 INCREMENT CRT DISPLAY (DI)	130							LIGHT PEN FLAG †	EDGE FLAG †	STOP FLAG †	DONE FLAG † DATA PI †	INITIALIZE	BITS 25-27 SPECIAL PI	BITS 30-32 SPECIAL PRIORITY INTERRUPT ASSIGNMENT			BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			2x18	COORDINATES TO COMPUTER
551 MICROTAPE CONTROL (UT)	210		UNIT SELECT ENABLE	ENABLE TAPE END FLAG	ENABLE JOB DONE FLAG	TURN ON/OFF SELECTED UNIT	MOVE FORWARD/REVERSE	ENABLE TIME FLAG	BITS 25-26 INITIAL TIME DELAY (0, 20, 150, 300 MS START DELAY)		BITS 27-29 CONTROL FUNCTION			BITS 30-32 UNIT SELECTION			BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			
551 STATUS REGISTER (CLEARED BY CONO 210)	214							DELAY IN PROGRESS	REQUEST	ACTIVE	NULL	INCOMPLETE BLOCK TRANSFER FLAG	BITS 28-26 TEST CONDITIONS (STATE OF CONTROL)	WRITING ON	TIME FLAG	PARITY ERROR	ILLEGAL OP FLAG	TAPE END FLAG	JOB DONE FLAG		
516 MAGNETIC TAPE CONTROL	220	FALSE END OF RECORD	WRITE CLOCK ENABLE	SLICE LEVEL 0/1	PARITY (EVEN/ODD) 0-BCD/1-BINARY	BITS 22-23 DENSITY SELECT 200/556 800/556 BPI		BITS 24, 25-27: FUNCTION 25-27: NO-OP, REWIND, W EOF, WRITE, READ COMP, READ, SPACE FWD, SPACE REV 24: SPECIAL FOR FWD UNLOAD WRITE FULL BLANK TAPE				HOLD SELECTED	BITS 29-31 UNIT SELECT (0-7)			MT DISABLE EOR TEST	BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT				
516 STATUS REGISTER (1)	224	ILLEGAL FLAG	136 DATA CONTROL CONNECTED	CONTINUE MOTION SELECTED	TRANSFER NEW COMMAND TO COMMAND HOLD BUFFER	TRANSPORT REWD	TAPE AT END POINT	TAPE AT LOAD POINT	SEE NOTE A ‡	SEE NOTE B ‡	END OF FILE FLAG	FILE PROTECT RING OUT	TAPE MISSED CHAR	READ COMPARE ERROR	TAPE PARITY ERROR (LONG)	TAPE PARITY ERROR (LAT)	END OF RECORD FLAG	TAPE TRANSPORT READY	TAPE CONTROL FREE		
516 STATUS REGISTER (2)	230	TRANSPORT UP TO SPEED DELAY IN PROGRESS	RECORD IN PROGRESS	END OF RECORD MOTION DELAY IN PROGRESS	MOTION STOP DELAY IN PROGRESS	COMMAND HOLD BUFFER	UNIT SELECTED NEW COMMAND	BITS 24-27 FUNCTION FINAL BUFFER				MOTION DELAY IN PROGRESS	BITS 29-31 UNIT FINAL BUFFER			FIRST CHAR DETECTED	EOR READ DELAY	MISSED CHAR DELAY	ECR WRITE DELAY		
630 DATA COMMUNICATION SYSTEM (DCS) CONTROL REGISTER	300															FLAG	BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			8 / 8	INPUT CHAR (SPECIFIED BY COUNTER), CLEAR UNIT FLAG / OUTPUT CHAR, CLEAR UNIT FLAG, USE BUF TO SELECT LINE RELEASE COUNTER
630: LINE COUNTER	304																BITS 33-35 PRIORITY INTERRUPT ASSIGNMENT			8 / 8	INPUT CHAR (SPECIFIED BY COUNTER), CLEAR UNIT FLAG, RELEASE COUNTER / OUTPUT CHAR, CLEAR UNIT FLAG, USE BUF TO SELECT LINE RELEASE COUNTER

NOTE:
BITS WHICH CAN'T BE READ OUT
BITS WHICH CAN'T BE READ IN
†(DAGGER) PI (PRIORITY OR PROGRAM BITS INTERRUPT OR CHANNEL)

†NOTE A: TAPE NEAR END POINT (520 INTERFACE) FIRST OPERATION WRITE (521 OR 522 INTERFACE)
NOTE B: TAPE NEAR LOAD POINT IF 520 INTERFACE B CONTROL USING TRANSPORT (521 INTERFACE) SEE POOL WRITE ECHO NOT OK (522 INTERFACE)

INPUT/OUTPUT SUMMARY CHART — PDP6

digital