# PROGRAMMED DATA PROCESSOR-8 USERS HANDBOOK

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Copyright 1966 Digital Equipment Corporation

# PREFACE

This handbook concerns programming, operating, and interfacing the Programmed Data Processor-8; a high-speed stored-program digital computer manufactured by the Digital Equipment Corporation. Section A describes the functional operation, instructions, and basic machine-language programming of the PDP-8 processor, core memory, processor options, and core memory options. Section B is devoted to the functional operation, instructions, and basic programming of standard and optional input/output equipment of a PDP-8 system. Section C presents information on operating the basic system and its options. Section D serves as an interface and installation manual, and contains information on planning and implementing the design and installation of any electrical interface required to connect a special device into a PDP-8 system.

Appendixes at the end of this handbook provide detailed information which may be helpful in specific programming assignments. Although program examples are given in this document, no attempt has been made to teach programming techniques. The meaning and use of special characters employed in the programming examples are explained in the description of the Program Assembly Language, available from the Digital Program Library.

# CONTENTS


Page

System Introduction ..... xiii
Computer Organization ..... xiii
Symbols ..... xiv

## SECTION A MEMORY AND PROCESSOR

Chapter 1 Memory and Processor Functional Description ..... 2
Major Registers ..... 2
Accumulator (AC) ..... 2
Link (L) ..... 3
Program Counter (PC) ..... 3
Memory Address Register (MA) ..... 3
Switch Register (SR) ..... 3
Core Memory ..... 3
Memory Buffer Register (MB) ..... 4
Instruction Register (IR) ..... 4
Major State Generator ..... 4
Output Bus Drivers ..... 5
Functional Summary ..... 7
Timing and Control Elements ..... 7
Timing Generators ..... 7
Register Controls ..... 7
Program Controls ..... 7
Interface ..... 8

Chapter 2 Memory and Processor Instructions ..... 10
Memory Reference Instructions ..... 10
Augmented Instructions ..... 12
Input Output Transfer Instruction ..... 12
Operate Instruction ..... 13

Chapter 3 Memory and Processor Basic Programming ..... 21
Memory Addressing ..... 21
Indirect Addressing ..... 23
Auto-Indexing ..... 23
Storing and Loading ..... 23
Program Control ..... 23
Indexing Operations ..... 23
Logic Operations ..... 24
Logical AND ..... 24
Inclusive OR ..... 24
Exclusive OR ..... 25
Arithmetic Operations ..... 25
Two's Complement Arithmetic ..... 25
Addition ..... 26
Subtraction ..... 26

Chapter 4 Program Interrupt ..... 27
Instructions ..... 27
Programming ..... 27

## CONTENTS (continued)


Page

Chapter 5 Data Break ..... 29
Single-Cycle Data Break ..... 31
Three-Cycle Data Break ..... 31
Word Count State ..... 32
Current Address State ..... 32
Break State ..... 32

Chapter 6 Memory Extension Control Type 183 and Memory Module Type 184 .. 33
Instructions ..... 35
Programming ..... 36

Chapter 7 Memory Parity Type 188 ..... 39
Instructions ..... 39
Programming ..... 40

Chapter 8 Extended Arithmetic Element Type 182 ..... 41
Instructions ..... 41
Programming ..... 45
Multiplication ..... 45
Division ..... 46

Chapter 9 Automatic Restart Type KR01 ..... 48


## SECTION B INPUT OUTPUT EQUIPMENT


Chapter 1 Basic IOT Programming ..... 52
Programmed Data Transfers ..... 52
Sense for Device Ready ..... 53
Assemble Data ..... 53
Effect a Transfer ..... 53
Program Interrupt ..... 54
Data Break Transfers ..... 55

Chapter 2 Teletype and Control ..... 57
Teletype Model 33 ASR ..... 57
Teletype Control ..... 57
Keyboard/Reader ..... 58
Teleprinter/Punch ..... 59
Teletype System Type LT08 ..... 61

Chapter 3 High Speed Perforated Tape Reader and Control Type 750C ..... 62

Chapter 4 High Speed Perforated Tape Punch and Control Type 75E ..... 64

Chapter 5 Analog-to-Digital Converter Type 189 ..... 66

Chapter 6 Analog-to-Digital Converter Type 138E and Multiplexer Control Type 139E ..... 68

Chapter 7 Digital-to-Analog Converter Type AA01 ..... 72

## CONTENTS (continued)


Page

Chapter 8 Display Equipment ... 73
Oscilloscope Display Type 34D ... 73
Precision CRT Display Type 30N ... 76
Light Pen Type 370 ... 76

Chapter 9 Incremental Plotter and Control Type 350B ... 78

Chapter 10 Card Reader and Control Type CR01C ... 81

Chapter 11 Card Reader and Control Type 451 ... 85

Chapter 12 Card Punch and Control Type 450 ... 87

Chapter 13 Automatic Line Printer and Control Type 645 ... 90

Chapter 14 Serial Magnetic Drum System Type 251 ... 94
Instructions ... 94
Programming ... 97

Chapter 15 DECtape Systems ... 99
DECtape Format ... 99
DECtape Dual Transport Type 555 and DECtape Control Type 552 ... 101
Instructions ... 104
Control Modes ... 106
Programmed Operation ... 106
DECtape Transport Type TU55 and DECtape Control Type TC01 ... 108
DECtape Transport Type TU55 ... 108
DECtape Control Type TC01 ... 109
Instructions ... 110
Control Modes ... 112
Functions ... 112
Programmed Operation ... 113
Software ... 116

Chapter 16 Automatic Magnetic Tape Control Type 57A ... 118
Functional Description ... 118
Instructions ... 119
Programming ... 125

Chapter 17 Magnetic Tape System Type 580 ... 128

Chapter 18 Data Communication Systems Type 680 ... 133
Data Line Interface Type 681 ... 135
Serial Line Multiplexer Type 685 ... 137
Instructions ... 137
Software ... 141


## SECTION C OPERATION


Chapter 1 Standard PDP-8 Operation ... 144
Controls and Indicators ... 144
Operating Procedures ... 149
Manual Data Storage and Modification ... 149
Loading Data Under Program Control ... 149
Off-Line Teletype Operation ... 150

## CONTENTS (continued)

Page

## SECTION D INTERFACE AND INSTALLATION


Chapter 1 PDP-8 Input/Output Facilities ..... 154
Programmed Data Transfers ..... 155
Data Break Transfers ..... 155
Logic Symbols ..... 155

Chapter 2 Programmed Data Transfers ..... 157
Timing and IOP Generator ..... 159
Device Selector (DS) ..... 161
Input/Output Skip (IOS) ..... 162
Accumulator ..... 163
Input Data Transfers ..... 164
Output Data Transfers ..... 165
Program Interrupt (PI) ..... 167
Multiple Use of IOS and PI ..... 168

Chapter 3 Data Break Transfers ..... 170
Single-Cycle Data Breaks ..... 171
Input Data Transfer ..... 171
Output Data Transfer ..... 173
Memory Increment ..... 175
Three-Cycle Data Breaks ..... 178

Chapter 4 Digital Logic Circuits ..... 181
Basic Digital Circuits ..... 181
Inverters ..... 181
Diode Gates ..... 182
Diode-Capacitor-Diode Gates ..... 184
Pulse Amplifiers ..... 185
Bus Drivers ..... 186
Interface Circuits of the Computer ..... 186
A502 Difference Amplifier ..... 187
R123 Diode Gate ..... 187
R210 PDP-8 Accumulator ..... 187
R211 MA, MB, and PC ..... 187
R650 Bus Driver ..... 188
S107 Inverter ..... 188
S111 Diode Gate ..... 189
S151 Binary-to-Octal Decoder ..... 189
S203 Triple Flip-Flop ..... 189
S603 Pulse Amplifier ..... 190
W640 Pulse Amplifier ..... 190
Interface Circuits of Peripheral Equipment ..... 190
W103 Device Selector ..... 191
R123 Diode Gate ..... 192

Chapter 5 Interface Connections ..... 193
Miscellaneous Interface Signals ..... 199
Address Extension Inputs and Data Fields Outputs ..... 199
Analog Input Signal ..... 199
B Run Output Signal ..... 199
BT1 and BT2A Output Pulses ..... 199
B Power Clear Output Pulses ..... 200

## CONTENTS (continued)


Page

Loading and Driving Considerations ..... 200
Base Load ..... 201
Pulse Load ..... 201
Pulsed Emitter Load ..... 202
DC Emitter Load ..... 202

Chapter 6 Installation Planning ..... 203
Space Requirements ..... 203
Environmental Requirements ..... 206
Power Requirements ..... 206
Cable Requirements ..... 206
Installation Procedure ..... 206

Appendix 1 Instructions ..... 210
Memory Reference Instructions ..... 210
Basic IOT Microinstructions ..... 212
Group 1 Operate Microinstructions ..... 224
Group 2 Operate Microinstructions ..... 225
Extended Arithmetic Element Microinstructions ..... 226

Appendix 2 Codes ..... 229
Model 33 ASR/KSR Teletype Code (ASCII) in Octal Form ..... 229
Model 33 ASR/KSR Teletype Code (ASCII) in Binary Form ..... 230
Card Reader Code ..... 231
Line Printer Code ..... 232

Appendix 3 Scales of Notation ..... 233

Appendix 4 Powers of Two ..... 234

Appendix 5 Octal - Decimal Conversion ..... 235
Octal - Decimal Integer Conversion Table ..... 235
Octal - Decimal Fraction Conversion Table ..... 239

Appendix 6 Perforated-Tape Loader Programs ..... 242
Readin Mode Loader ..... 242
Binary Loader ..... 243

Appendix 7 Programming System ..... 245
Featured Programs ..... 245
Abstracts of Programs ..... 247


## ILLUSTRATIONS


Figure 1 Typical PDP-8 in Table-Top Configuration ..... xii
Figure 2 PDP-8 Major Register Block Diagram ..... 2
Figure 3 PDP-8 Timing and Control Element Block Diagram ..... 6
Figure 4 Memory Reference Instruction Bit Assignments ..... 10
Figure 5 IOT Instruction Bit Assignments ..... 13
Figure 6 Group 1 Operate Microinstruction Bit Assignments ..... 13
Figure 7 Group 2 Operate Microinstruction Bit Assignments ..... 17
Figure 8 EAE Microinstruction Bit Assignments ..... 41
Figure 9 DECtape Track Allocation ..... 100
Figure 10 DECtape Mark Channel Format ..... 100

## ILLUSTRATIONS (continued)


| | | Page |
|---|---|---|
| Figure 11 | DECtape Control Word and Data Word Assignments | 100 |
| Figure 12 | DECtape Format Details | 100 |
| Figure 13 | DECtape Status Register A Bit Assignments | 114 |
| Figure 14 | DECtape Status Register B Bit Assignments | 114 |
| Figure 15 | Data Communication System Block Diagram | 133 |
| Figure 16 | Typical Teletype Line Timing | 134 |
| Figure 17 | Teletype In Instruction Flow Diagram | 136 |
| Figure 18 | PDP-8 Operator Console | 144 |
| Figure 19 | Teletype Model 33 ASR Console | 148 |
| Figure 20 | Digital Logic Symbols | 155 |
| Figure 21 | Programmed Data Transfer Interface Block Diagram | 158 |
| Figure 22 | Programmed Data Transfer Timing | 158 |
| Figure 23 | Typical IOT Instruction Decoding | 159 |
| Figure 24 | IOP Generator Logic | 160 |
| Figure 25 | Generation of IOT Command Pulses By Device Selectors | 161 |
| Figure 26 | Typical Device Selector (Device 34) | 162 |
| Figure 27 | Use of IOS to Test the Status of an External Device | 163 |
| Figure 28 | Accumulator Input and Output | 164 |
| Figure 29 | Loading Data into the Accumulator from an External Device | 165 |
| Figure 30 | Loading a 6-Bit Word into an External Device from the Accumulator | 166 |
| Figure 31 | Use of a Device Selector for Activating and Controlling an External Device | 166 |
| Figure 32 | Program Interrupt Request Signal Origin | 167 |
| Figure 33 | Multiple Inputs to IOS and PI Facilities | 168 |
| Figure 34 | Data Break Transfer Interface Block Diagram | 170 |
| Figure 35 | Single-Cycle Data Break Input Transfer Timing Diagram | 171 |
| Figure 36 | Device Interface Logic for Single-Cycle Data Break Input Transfer | 172 |
| Figure 37 | Single-Cycle Data Break Output Transfer Timing Diagram | 173 |
| Figure 38 | Device Interface Logic for Single-Cycle Data Break Output Transfer | 174 |
| Figure 39 | Device Interface Logic for Strobing Output Data | 175 |
| Figure 40 | Memory Increment Data Break Timing Diagram | 176 |
| Figure 41 | Device Interface Logic for Memory Increment Data Break | 177 |
| Figure 42 | Three-Cycle Data Break Timing Diagram | 179 |
| Figure 43 | Inverter Circuit Schematic Diagram | 181 |
| Figure 44 | Direct-Set Input Circuit Schematic of the R210 PDP-8 Accumulator | 182 |
| Figure 45 | Single-Input Diode Gate Circuit Schematic | 182 |
| Figure 46 | Multiple-Input Diode Gate Circuit Schematic | 183 |
| Figure 47 | Parallel-Connected Diode Gate Circuit Schematic | 183 |
| Figure 48 | Diode Gate Logic Symbol | 184 |
| Figure 49 | Logic Operations Performed by Diode Gates | 184 |
| Figure 50 | Diode-Capacitor-Diode Gate Circuit Schematic | 185 |
| Figure 51 | Diode-Capacitor-Diode Gate Logic Symbol | 185 |
| Figure 52 | Parallel-Connected Trigger Pulse to DCD Gates | 185 |
| Figure 53 | Pulse Amplifier Output Circuit Schematic | 186 |
| Figure 54 | Bus Driver Output Circuit Schematic | 186 |
| Figure 55 | Device Selector W103 Logic Circuit | 191 |
| Figure 56 | Diode Gate R123 Logic Circuit | 192 |
| Figure 57 | Table Mounted PDP-8 Installation Dimensions | 203 |
| Figure 58 | Cabinet Mounted PDP-8 Installation Dimensions | 204 |
| Figure 59 | Optional Cabinet and Table Installation Dimensions | 205 |
| Figure 60 | Typical PDP-8 System Configuration and Layout Planning | 209 |

# SECTION A

# MEMORY AND PROCESSOR

# TABLES


| | | Page |
|---|---|---|
| Table 1 | Analog-to-Digital Converter Type 189 Characteristics | 67 |
| Table 2 | Analog-to-Digital Converter Type 138E Characteristics | 68 |
| Table 3 | Operator Console Controls and Indicators | 144 |
| Table 4 | Teletype Controls and Indicators | 148 |
| Table 5 | Programmed Data Transfer Input Signals | 194 |
| Table 6 | Programmed Data Transfer Output Signals | 195 |
| Table 7 | Data Break Transfer Input Signals | 196 |
| Table 8 | Data Break Transfer Output Signals | 197 |
| Table 9 | Miscellaneous Input Signals | 198 |
| Table 10 | Miscellaneous Output Signals | 198 |
| Table 11 | Installation Data | 207 |
| Table 12 | Space Requirements | 208 |

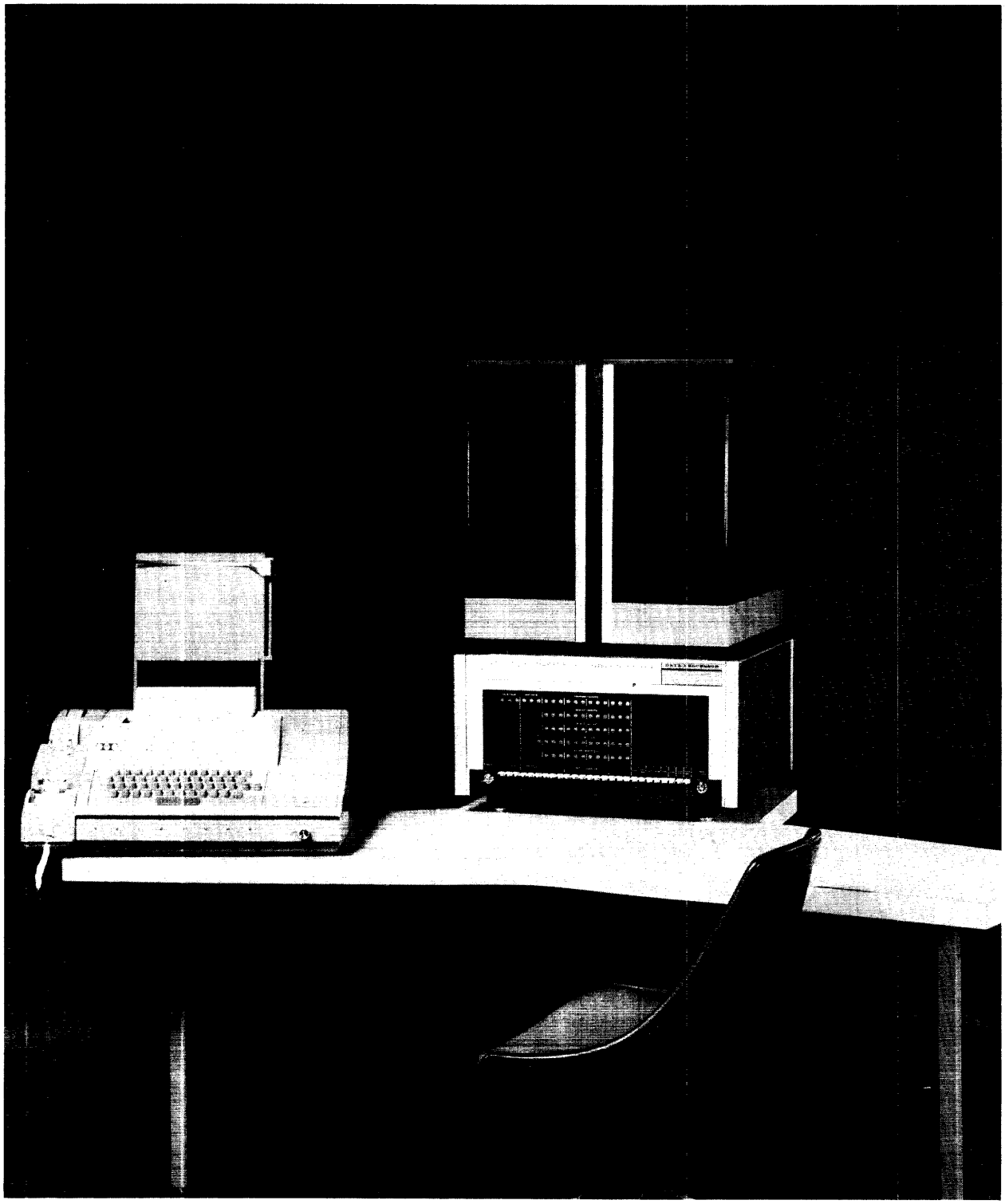


Figure 1 Typical PDP-8 in Table Top Configuration

# SYSTEM INTRODUCTION

The Digital Equipment Corporation Programmed Data Processor-8 (PDP-8) is designed for use as a small-scale general-purpose computer, an independent information handling facility in a larger computer system, or as the control element in a complex processing system. The PDP-8 is a one-address, fixed word length, parallel computer using 12 bit, two's complement arithmetic. Cycle time of the 4096-word random-address magnetic-core memory is 1.5 microseconds. Standard features of the system include indirect addressing and facilities for instruction skipping and program interruption as functions of input-output device conditions.

The 1.5-microsecond cycle time of the machine provides a computation rate of 333,333 additions per second. Addition is performed in 3.0 microseconds (with one number in the accumulator) and subtraction is performed in 6.0 microseconds (with the subtrahend in the accumulator). Multiplication is performed in approximately 315 microseconds by a subroutine that operates on two signed 12-bit numbers to produce a 24-bit product, leaving the 12 most significant bits in the accumulator. Division of two signed 12-bit numbers is performed in approximately 444 microseconds by a subroutine that produces a 12-bit quotient in the accumulator and a 12-bit remainder in core memory. Similar multiplication and division operations are performed by means of the optional extended arithmetic element in approximately 15 and 30 microseconds, respectively.

Flexible, high-capacity, input-output capabilities of the computer allow it to operate a variety of peripheral equipment. In addition to standard Teletype and perforated tape equipment, the system is capable of operating in conjunction with a number of optional devices such as high-speed perforated tape readers and punches, card equipment, a line printer, analog-to-digital converters, cathode-ray-tube displays, magnetic drum systems, and magnetic-tape equipment. Equipment of special design is easily adapted for connection into the PDP-8 system. The computer is not modified with the addition of peripheral devices.

PDP-8 is completely self-contained, requiring no special power sources or environmental conditions. A single source of 115-volt, 60-cycle, single-phase power is required to operate the machine. Internal power supplies produce all of the operating voltages required. FLIP CHIP modules utilizing hybrid silicon circuits and built-in provisions for marginal checking insure reliable operation in ambient temperatures between 32 and 130 degrees Fahrenheit.

## Computer Organization

The PDP-8 system is organized into a processor, core memory, and input/output equipment and facilities. All arithmetic, logic, and system control operations of the standard PDP-8 are performed by the processor. Permanent (longer than one instruction time) local information storage and retrieval operations are performed by the core memory. The memory is continuously cycling, automatically performing a read and write operation during each computer cycle. Input and output address and data buffering for the core memory is performed by registers of the processor, and operation of the memory is under control of timing signals produced by the processor. Due to the close relationship of operations performed by the processor and the core memory, these two elements are described together in Section A of this handbook.

Interface circuits for the processor allow bussed connections to a variety of peripheral equipment. Each input/output device is responsible for detecting its own select code and for providing any necessary input or output gating. Individually programmed data transfers between the processor and peripheral equipment take place through the processor accumulator. Data transfers can be initiated by peripheral equipment, rather

than by the program, by means of the data break facilities. Standard features of the PDP-8 also allow peripheral equipment to perform certain control functions such as instruction skipping, and a transfer of program control initiated by a program interrupt.

Standard peripheral equipment provided with each PDP-8 system consists of a Teletype Model 33 Automatic Send Receive set and a Teletype control. The Teletype unit is a standard machine operating from serial 11-unit-code characters at a rate of ten characters per second. The Teletype provides a means of supplying data to the computer from perforated tape or by means of a keyboard, and supplies data as an output from the computer in the form of perforated tape or typed copy. The Teletype control serves as a serial-to-parallel converter for Teletype inputs to the computer and serves as a parallel-to-serial converter for computer output signals to the Teletype unit.

The Teletype and all optional input/output equipment is discussed in Section B of this handbook.

# Symbols

The following special symbols are used throughout this handbook to explain the function of equipment and instructions:

| Symbol | Explanation |
|---|---|
| A => B | The content of register A is transferred into register B |
| 0 => A | Register A is cleared to contain all binary zeros |
| Aj | Any given bit in A |
| A5 | The content of bit 5 of register A |
| A5(1) | Bit 5 of register A contains a 1 |
| A6 - 11 | The content of bits 6 through 11 of register A |
| A6 - 11 => B0 - 5 | The content of bits 6 through 11 of register A is transferred into bits 0 through 5 of register B |
| Y | The content of any core memory location |
| V | Inclusive OR |
| $\not\vee$ | Exclusive OR |
| $\wedge$ | AND |
| $\overline{A}$ | Ones complement of the content of A |

# CHAPTER 1

# MEMORY AND PROCESSOR FUNCTIONAL DESCRIPTION

## Major Registers

To store, retrieve, control, and modify information and to perform the required logical, arithmetic, and data processing operations, the core memory and the processor employ the logic components shown in Figure 2 and described in the following paragraphs.

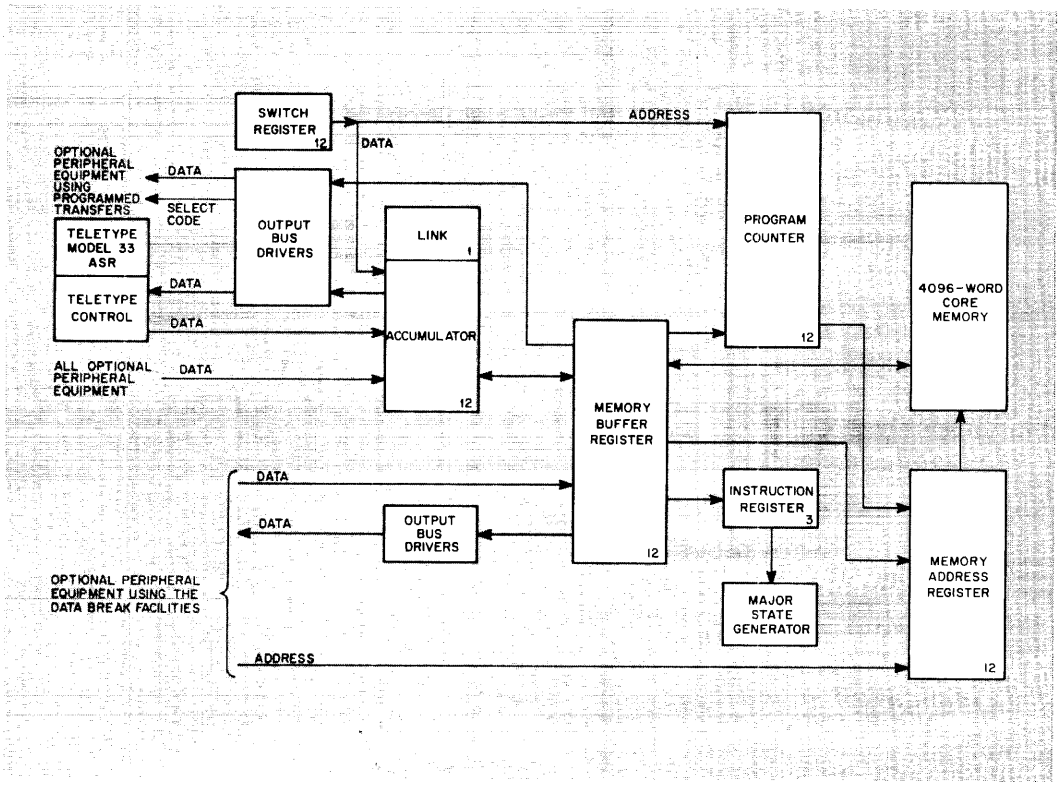



Figure 2 PDP-8 Major Register Block Diagram

## ACCUMULATOR (AC)

Arithmetic and logic operations are performed in this 12-bit register. Under program control the AC can be cleared or complemented, its content can be rotated right or left with the link. The content of the memory buffer register can be added to the content of the AC and the result left in the AC. The content of both of these registers may be combined by the logical operation AND, the result remaining in the AC. The memory buffer register and the AC also have gates which allow them to be used together as the shift register and buffer register of a successive approximation analog-to-digital converter. The inclusive OR may be performed between the AC and the switch register on the operator console and the result left in the AC.

The accumulator also serves as an input-output register. All programmed information transfers between core memory and an external device pass through the accumulator.

## LINK (L)

This one-bit register is used to extend the arithmetic facilities of the accumulator. It is used as the carry register for two's complement arithmetic. Overflow into the L from the AC can be checked by the program to greatly simplify and speed up single and multiple precision arithmetic routines. Under program control the link can be cleared and complemented, and it can be rotated as part of the accumulator.

## PROGRAM COUNTER (PC)

The program sequence, that is the order in which instructions are performed, is determined by the PC. This 12-bit register contains the address of the core memory location from which the next instruction will be taken. Information enters the PC from the core memory, via the memory buffer register, and from the switch register on the operator console. Information in the PC is transferred into the memory address register to determine the core memory address from which each instruction is taken. Incrementation of the content of the PC establishes the successive core memory locations of the program and provides skipping of an instruction based upon a programmed test of information or conditions.

## MEMORY ADDRESS REGISTER (MA)

The address in core memory which is currently selected for reading or writing is contained in this 12-bit register. Therefore, all 4096 words of core memory can be addressed directly by this register. Data can be set into it from the memory buffer register, from the program counter, or from an I/O device using the data break facilities.

## SWITCH REGISTER (SR)

Information can be manually set into the switch register for transfer into the PC as an address by means of the LOAD ADDRESS key, or into the AC as data to be stored in core memory by means of the DEPOSIT key.

## CORE MEMORY

The core memory provides storage for instructions to be performed and information to be processed or distributed. This random address magnetic core memory holds 4096 12-bit words in the standard PDP-8. Optional equipment extends the storage capacity in fields of 4096 words or expands the word length to 13 bits to provide parity checking. Memory location $0_8$ is used to store the content of the PC following a program interrupt, and location $1_8$ is used to store the first instruction to be executed following a program interrupt. (When a program interrupt occurs, the content of the PC is stored in location $0_8$, and program control is transferred to location 1 automatically.) Locations $10_8$ through $17_8$ are used for auto-indexing. All other locations can be used to store instructions or data.

Core memory contains numerous circuits such as read-write switches, address decoders, inhibit drivers, and sense amplifiers. These circuits perform the electrical conversions necessary to transfer information into or out of the core array and perform no arithmetic or logic operations upon the data. Since their operation is not discernible by the programmer or operator of the PDP-8, these circuits are not described here in detail.

## MEMORY BUFFER REGISTER (MB)

All information transfers between the processor registers and the core memory are temporarily held in the MB. Information can be transferred into the MB from the accumulator or memory address register. The MB can be cleared, incremented by one or two, or shifted right. Information can be set into the MB from an external device during a data break or form core memory, via the sense amplifiers. Information is read from a memory location in 0.8 microsecond and rewritten in the same location in another 0.8 microsecond of one 1.6-microsecond memory cycle.

## INSTRUCTION REGISTER (IR)

This 3-bit register contains the operation code of the instruction currently being performed by the machine. The three most significant bits of the current instruction are loaded into the IR from the memory buffer register during a Fetch cycle. The content of the IR is decoded to produce the eight basic instructions, and affect the cycles and states entered at each step in the program.

## MAJOR STATE GENERATOR

One or more major states are entered serially to execute programmed instructions or to effect a data break. The major state generator establishes one state for each computer timing cycle. The Fetch, Defer, and Execute states are entered to determine and execute instructions. Entry into these states is produced as a function of the current instruction and the current state. The Word Count, Current Address, and Break states are entered during a data break. The Break state or all three of these states are entered based upon request signals received from peripheral I/O equipment.

### Fetch

During this state an instruction is read into the MB from core memory at the address specified by the content of the PC. The instruction is restored in core memory and retained in the MB. The operation code of the instruction is transferred into the IR to cause enactment, and the content of the PC is incremented by one.

If a multiple-cycle instruction is fetched, the following major state will be either Defer or Execute. If a one-cycle instruction is fetched, the operations specified are performed during the last part of the Fetch cycle and the next state will be another Fetch.

### Defer

When a 1 is present in bit 3 of a memory reference instruction, the Defer state is entered to obtain the full 12-bit address of the operand from the address in the current page or page 0 specified by bits 4 through 11 of the instruction. The process of address deferring is called indirect addressing because access to the operand is addressed indirectly, or deferred, to another memory location.

## Execute

This state is entered for all memory reference instructions except jump. During an AND, two's complement add, or increment and skip if zero instruction, the content of the core memory location specified by the address portion of the instruction is read into the MB and the operation specified by bits 0 through 2 of the instruction is performed. During a deposit and clear accumulator instruction the content of the AC is transferred into the MB and is stored in core memory at the address specified in the instruction. During a jump to subroutine instruction this state occurs to write the content of the PC into the core memory address designated by the instruction and to transfer this address into the PC to change program control.

## Word Count

This state is entered when an external device supplies signals requesting a data break and specifying that the break should be a 3-cycle break. When this state occurs, a transfer word count in a core memory location designated by the device is read into the MB, is incremented by 1, and is rewritten in the same location. If the word count overflows, indicating that the desired number of data break transfers will be enacted at completion of the current break, the computer transmits a signal to the device. The Current Address state immediately follows the Word Count state.

## Current Address

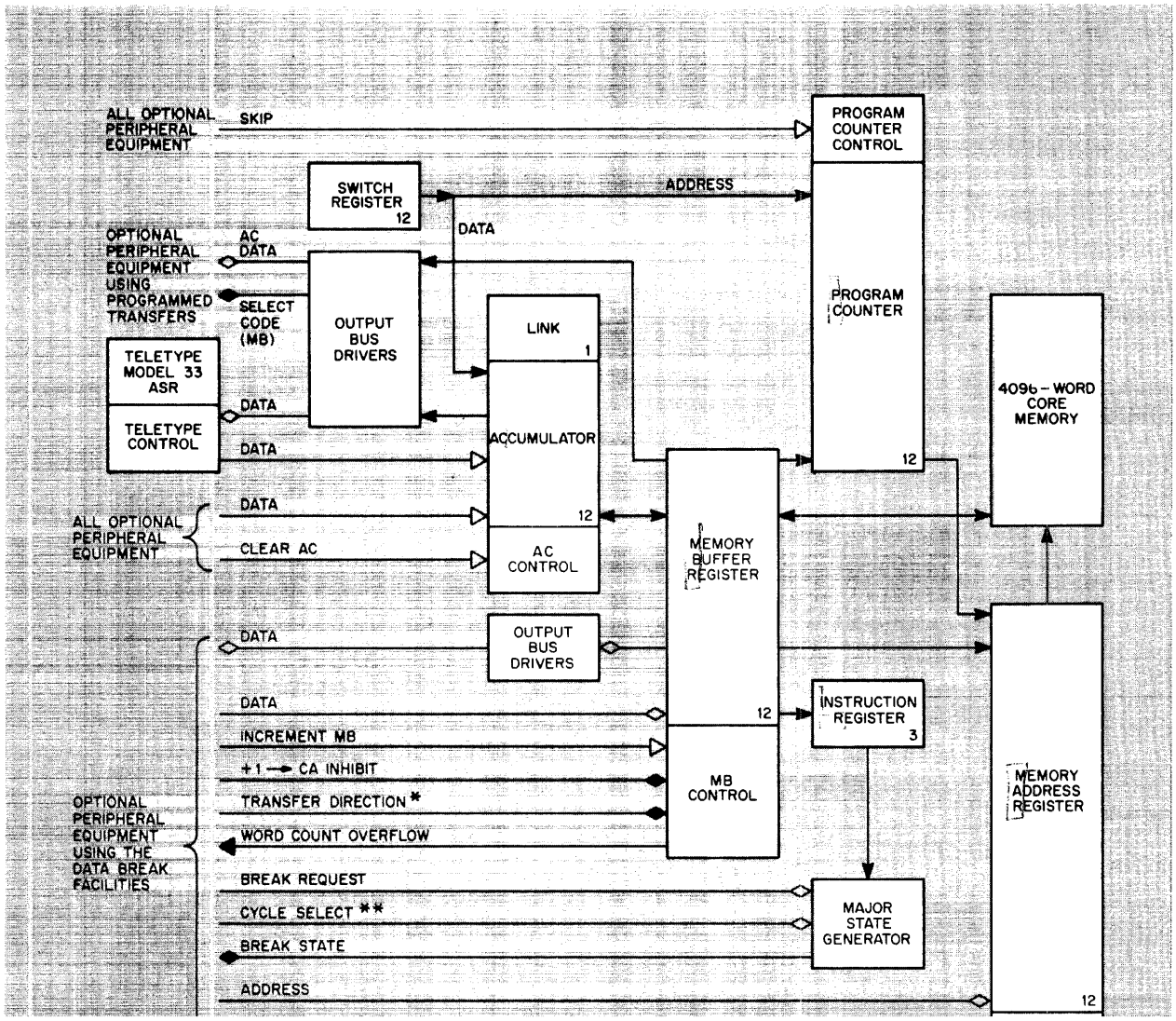
As the second cycle of a 3-cycle data break, this cycle establishes the address for the transfer that takes place in the following cycle (Break state). Normally the location following the word count is read from core memory into the MB, is incremented by 1 to establish sequential addresses for the transfers, and is transferred into the MA to determine the address selected for the next cycle. When the word count operation is not used, the device supplies an inhibit signal to the computer so that the word read during this cycle is not incremented. Transfers occur at sequential addresses due to incrementing during the Word Count state. During this sequence the word in the MB is rewritten at the same location and the MB is cleared at the end of the cycle. The Break state immediately follows the Current Address state.

## Break

This state is entered to enact a data transfer between computer core memory and an external device, either as the only state of a 1-cycle data break or as the final state of a 3-cycle data break. When a break request signal arrives and the cycle select signal specifies a 1-cycle break, the computer enters the Break state at the completion of the current instruction. Information transfers occur between the external device and a device-specified core memory location, through the MB. When this transfer is complete, the program sequence resumes from the point of the break. The data break does not affect the content of the AC, L, and PC.

# OUTPUT BUS DRIVERS

Output signals from the computer processor are power amplified by output bus driver modules of the standard PDP-8; allowing these signals to drive a heavy circuit load.


ALL OPTIONAL PERIPHERAL EQUIPMENT
SKIP
PROGRAM COUNTER CONTROL
SWITCH REGISTER 12
ADDRESS
DATA
OPTIONAL PERIPHERAL EQUIPMENT USING PROGRAMMED TRANSFERS
AC DATA
SELECT CODE (MB)
OUTPUT BUS DRIVERS
LINK 1
PROGRAM COUNTER 12
4096 – WORD CORE MEMORY
TELETYPE MODEL 33 ASR
TELETYPE CONTROL
DATA
ACCUMULATOR 12
DATA
ALL OPTIONAL PERIPHERAL EQUIPMENT
DATA
CLEAR AC
AC CONTROL
MEMORY BUFFER REGISTER
OUTPUT BUS DRIVERS
DATA
DATA
12
INSTRUCTION REGISTER 3
INCREMENT MB
+1 → CA INHIBIT
TRANSFER DIRECTION *
MB CONTROL
MEMORY ADDRESS REGISTER
OPTIONAL PERIPHERAL EQUIPMENT USING THE DATA BREAK FACILITIES
WORD COUNT OVERFLOW
BREAK REQUEST
CYCLE SELECT **
BREAK STATE
MAJOR STATE GENERATOR
ADDRESS
12

## FUNCTIONAL SUMMARY

Operation of the computer is accomplished on a limited scale by keys on the operator console. Operation in this manner is limited to address and data storage by means of the switch register, core memory data examination, the normal start/stop/continue control, and the single step or single instruction operation that allows a program to be monitored visually as a maintenance operation. Most of these manually initiated operations are performed by executing an instruction in the same manner as by automatic programming, except that the gating is performed by special pulses rather than by the normal clock pulses. In automatic operation, instructions stored in core memory are loaded into the memory buffer register and executed during one or more computer cycles. Each instruction determines the major control states that must be entered for its execution. Each control state lasts for one 1.5-microsecond computer cycle and is divided into distinct time states which can be used to perform sequential logical operations. Performance of any function of the computer is controlled by gating of a specific instruction during a specific major control state and a specific time state.

# Timing and Control Elements

The circuit elements that determine the timing and control, of the operation of the major registers of the PDP-8 are added to Figure 2 to form Figure 3. Figure 3 shows the timing and control elements described in the succeeding paragraphs and indicates their relationship to the major registers. These elements can be grouped categorically into timing generators, register controls, and program controls.

## TIMING GENERATORS

Timing pulses used to determine the computer cycle time and used to initiate sequential time-synchronized gating operations are produced by the timing signal generator. Timing pulses used during operations resulting from the use of the keys and switches on the operator console are produced by the special pulse generator. Pulses that reset registers and control circuits during power turn on and turn off operations are produced by the power clear pulse generator. Several of these pulses are available to peripheral devices using programmed or data break information transfers.

## REGISTER CONTROLS

Operation of the AC, MA, MB, and PC is controlled by an associated logic circuit. These circuits, in turn, transmit and receive control signals to and from I/O equipment. Programmed data transfer equipment can supply a pulse to the AC control to clear the AC prior to a data input and can supply a pulse to cause the content of the PC to be incremented, thus initiating an instruction skip. Equipment using the data break facility passes signals with the MA control and MB control to determine the direction and timing of data transfers in this mode.

## PROGRAM CONTROLS

Circuits are also included in the PDP-8 that produce the IOP pulses which initiate operations involved in input/output transfers, determine the advance of the computer program, and allow peripheral equipment to cause a program interrupt of the main computer program to transfer program control to a subroutine which performs some service for the I/O device.

# Interface

The input/output portion of the PDP-8 is extremely flexible and interfaces readily with special equipment, especially in real time data processing and control environments.

The PDP-8 utilizes a "bus" I/O system rather than the more conventional "radial" system. The "bus" system allows a single set of data and control lines to communicate with all I/O devices. The bus simply goes from one device to the next. No additional connections to the computer are required. A "radial" system requires that a different set of signals be transmitted to each device; and thus the computer must be modified when new devices are added. The PDP-8 need not be modified when adding new peripheral devices.

External devices receive two types of information from the computer: data and control signals. Computer output data is present as static levels on 12 lines. These levels represent a 12-bit word to be transmitted in parallel to a device. Data signals are received at all devices but are sampled only by the appropriate one in response to a control signal. Control signals are of two types: levels and timing pulses. Six static levels and their complement are supplied by the MB on 12 lines. These lines contain a code representing the device from which action is required. Each device recognizes its own code and performs its function only when this code is present. There are three timing pulses which may be programmed to occur. These IOP pulses are separated in time by one microsecond and are brought to all devices on 3 lines. These pulses are used by a device only when it is selected by the appropriate code on the level lines. They may be used to perform sequential functions in an external register, such as clear and read, or any other function requiring one, two, or three sequential pulses.

Peripheral devices transmit information to the computer on four types of "busses". These are the information bus, the clear AC bus, the skip bus, and the program interrupt bus. The information bus consists of 12 lines normally held at −3 volts by load resistors within the computer. Whenever one of these lines is brought to ground, a binary 1 will be placed in the corresponding accumulator bit. Each device may use the input bus only when it is selected; and thus, these input lines are time shared among all of the connected devices. The skip bus is electrically identical to the information bus. However, when it is driven to ground the next sequential instruction will be skipped. It too can be used only by the device currently selected and is effectively time shared. The program interrupt bus may be driven to ground at any time by any device whether currently selected or not. When more than one device is connected to the interrupt bus they should also be connected to the skip bus so the program can identify the device requesting program interruption.

The transmission of device selection levels and timing pulses is completely under program control. A single instruction can select any one of 64 devices and transmit up to three IOP timing pulses. Since the timing pulses are individually programmable, one might be used to strobe data into an external device buffer, another to transmit data to the computer, and the third to test a status flip-flop and drive the skip bus to ground if it is in the enabling state.

Data transfers may also be made directly with core memory at a high speed using the data break facility. This is a completely separate I/O system from the one described previously. It is standard equipment in every PDP-8 and is ordinarily used with fast I/O devices such as magnetic drums or tapes. Transfers through the data break facility are interlaced with the program in progress. They are initiated by a request from the peripheral device and not by programmed instruction. Thus, the device may transfer a word with memory whenever it is ready and does not have to wait for the program to issue an instruction. Computation may proceed on an interlaced basis with these transfers.

Interface signal characteristics are indicated in Section D of this handbook.

# CHAPTER 2

# MEMORY AND PROCESSOR INSTRUCTIONS

Instruction words are of two types: memory reference and augmented. Memory reference instructions store or retrieve data from core memory, while augmented instructions do not. All instructions utilize bits 0 through 2 to specify the operation code. Operation codes of $0_8$ through $5_8$ specify memory reference instructions, and codes of $6_8$ and $7_8$ specify augmented instructions. Memory reference instruction execution times are multiples of the 1.5-microsecond memory cycle. Indirect addressing increases the execution time of a memory reference instruction by 1.5 microseconds. The augmented instructions, input-output transfer and operate, are performed in 3.75 and 1.5 microseconds respectively.

## Memory Reference Instructions

Since the PDP-8 system contains a 4096-word core memory, 12 bits are required to address all locations. To simplify addressing, the core memory is divided into blocks, or pages, of 128 words ($200_8$ addresses). Pages are numbered $0_8$ through $37_8$, each field of 4096-words of core memory uses 32 pages. The seven address bits (bits 5 through 11) of a memory reference instruction can address any location in the page on which the current instruction is located by placing a 1 in bit 4 of the instruction. By placing a 0 in bit 4 of the instruction, any location in page 0 can be addressed directly from any page of core memory. All other core memory locations can be addressed indirectly by placing a 1 in bit 3 and placing a 7-bit effective address in bits 5 through 11 of the instruction to specify the location in the current page or page 0 which contains the full 12-bit absolute address of the operand.

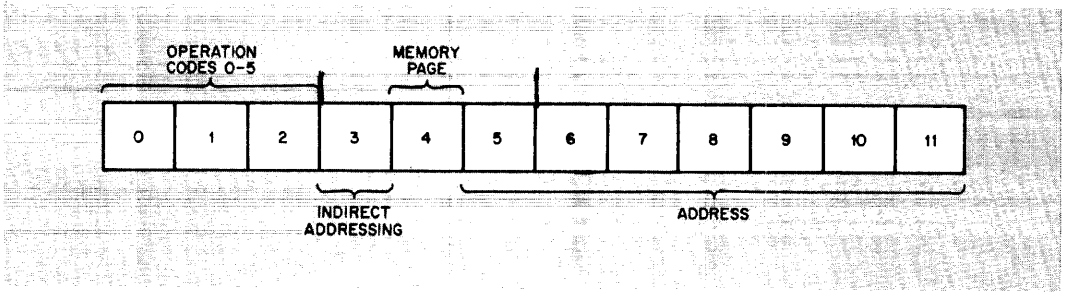



Figure 4 Memory Reference Instruction Bit Assignments

Word format of memory reference instructions is shown in Figure 4 and the instructions perform as follows:

LOGICAL AND (AND Y)

*Octal Code:* 0

*Indicators:* AND, FETCH, EXECUTE

*Execution Time:* 3.0 microseconds with direct addressing, 4.5 microseconds with indirect addressing.

*Operation:* The AND operation is performed between the content of memory location Y

and the content of the AC. The result is left in the AC, the original content of the AC is lost, and the content of Y is restored. Corresponding bits of the AC and Y are operated upon independently. This instruction, often called extract or mask, can be considered as a bit-by-bit multiplication. Example:

| Original ACj | Yj | Final ACj |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

*Symbol:* ACj $\wedge$ Yj $=>$ ACj

## TWO'S COMPLEMENT ADD (TAD Y)

*Octal Code:* 1

*Indicators:* TAD, FETCH, EXECUTE

*Execution Time:* 3.0 microseconds with direct addressing, 4.5 microseconds with indirect addressing.

*Operation:* The content of memory location Y is added to the content of the AC in two's complement arithmetic. The result of this addition is held in the AC, the original content of the AC is lost, and the content of Y is restored. If there is a carry from AC0, the link is complemented. This feature is useful in multiple precision arithmetic.

*Symbol:* AC0 $-$ 11 $+$ Y0 $-$ 11 $=>$ AC0 $-$11

## INCREMENT AND SKIP IF ZERO (ISZ Y)

*Octal Code:* 2

*Indicators:* ISZ, FETCH, EXECUTE

*Execution Time:* 3.0 microseconds with direct addressing, 4.5 microseconds with indirect addressing.

*Operation:* The content of memory location Y is incremented by one in two's complement arithmetic. If the resultant content of Y equals zero, the content of the PC is incremented by one and the next instruction is skipped. If the resultant content of Y does not equal zero, the program proceeds to the next instruction. The incremented content of Y is restored to memory. The content of the AC is not affected by this instruction.

*Symbol:* Y $+$ 1 $=>$ Y
If resultant Y0 $-$ 11 $=$ 0, then PC $+$ 1 $=>$ PC

## DEPOSIT AND CLEAR AC (DCA Y)

*Octal Code:* 3

*Indicators:* DCA, FETCH, EXECUTE

*Execution Time:* 3.0 microseconds with direct addressing, 4.5 microseconds with indirect addressing.

*Operation:* The content of the AC is deposited in core memory at address Y and the

AC is cleared. The previous content of memory location Y is lost.

*Symbol:* AC => Y
then O => AC

JUMP TO SUBROUTINE (JMS Y)

*Octal Code:* 4

*Indicators:* JMS, FETCH, EXECUTE

*Execution Time:* 3.0 microseconds with direct addressing, 4.5 microseconds with indirect addressing.

*Operation:* The content of the PC is deposited in core memory location Y and the next instruction is taken from core memory location Y + 1. The content of the AC is not affected by this instruction.

*Symbol:* PC + 1 => Y
Y + 1 => PC

JUMP TO Y (JMP Y)

*Octal Code:* 5

*Indicators:* JMP, FETCH

*Execution Time:* 1.5 microseconds with direct addressing, 3.0 microseconds with indirect addressing.

*Operation:* Address Y is set into the PC so that the next instruction is taken from core memory address Y. The original content of the PC is lost. The content of the AC is not affected by this instruction.

*Symbol:* Y => PC

# Augmented Instructions

There are two augmented instructions which do not reference core memory. They are the input-output transfer, which has an operation code of 6, and the operate which has an operation code of 7. Bits 3 through 11 within these instructions function as an extension of the operation code and can be microprogrammed to perform several operations within one instruction. Augmented instructions are one-cycle (Fetch) instructions that initiate various operations as a function of bit microprogramming. The operations initiated by each bit occur at a specified time with respect to the computer cycle time and are designated as event times 1, 2, and 3. Three event times, separated by 1 microsecond, occur during the input-output transfer instruction. Two event times occur during the 1.5-microsecond cycle time of an operate instruction.

## INPUT OUTPUT TRANSFER INSTRUCTION

Microinstructions of the input-output transfer (IOT) instruction initiate operation of peripheral equipment and effect information transfers between the processor and an I/O device. Specifically, when an operation code of 6 is detected, the PAUSE flip-flop is set and the IOP generator is enabled to produce IOP 1, IOP 2, and IOP 4 pulses as a function of the content of instruction bits 9 through 11. These pulses occur at 1-microsecond intervals designated as event times 3, 2, and 1 as follows:

| Instruction Bit | IOP Pulse | IOT Pulse | Event Time |
|---|---|---|---|
| 11 | IOP 1 | IOT 1 | 1 |
| 10 | IOP 2 | IOT 2 | 2 |
| 9 | IOP 4 | IOT 4 | 3 |

The IOP pulses are gated in the device selector of the program-selected equipment to produce IOT pulses that enact a data transfer or initiate a control operation. Selection of an equipment is accomplished by bits 3 through 8 of the IOT instruction. These bits form a 6-bit code that enables the device selector in a given device.

The format of the IOT instruction is shown in Figure 5. Operations performed by IOT microinstructions are explained in Section B of this handbook.

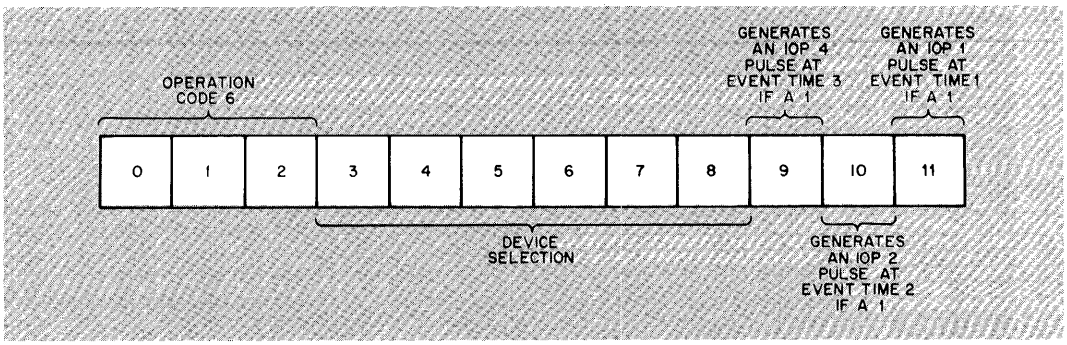



Figure 5 IOT Instruction Bit Assignments

## OPERATE INSTRUCTION

**The operate instruction consists of two groups of microinstructions. Group 1 (OPR 1) is principally for clear, complement, rotate, and increment operations and is designated by the presence of a 0 in bit 3. Group 2 (OPR 2) is used principally in checking the content of the accumulator and link and continuing to, or skipping, the next instruction based on the check. A 1 in bit 3 designates an OPR 2 microinstruction.**

Group 1 operate microinstruction format is shown in Figure 6 and the microinstructions are explained in the succeeding paragraphs. Any logical combination of bits within this group can be combined into one microinstruction. For example, it is possible to assign ones to bits 5, 6, and 11; but it is not logical to assign ones to bits 8 and 9 simultaneously since they specify conflicting operations. The only restriction on combining OPR 1 operations within one instruction, other than logical conflicts, is that a rotate operation (bits 8, 9, or 10) may not be combined with the increment AC operation (bit 11) since they are executed at the same event time.

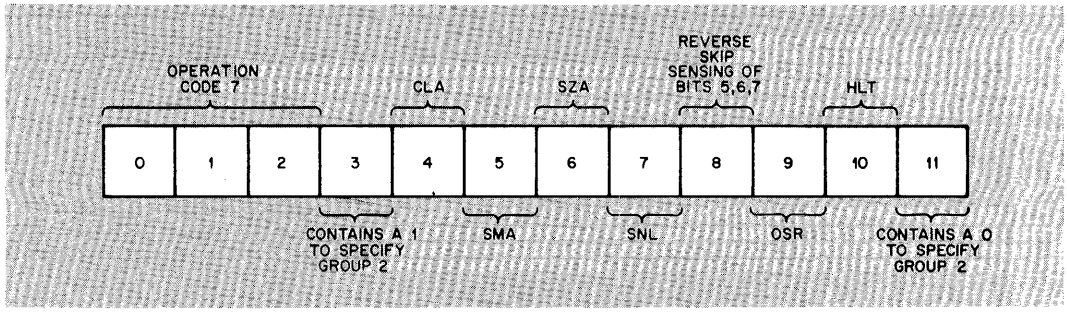



Figure 6 Group 1 Operate Instruction Bit Assignments

### NO OPERATION (NOP)

*Octal Code:* 7000

*Event Time:* None

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* This command causes a 1-cycle delay in the program and then the next sequential instruction is initiated. This command is used to add execution time to a program, such as to synchronize subroutine or loop timing with peripheral equipment timing.

*Symbol:* None

### INCREMENT ACCUMULATOR (IAC)

*Octal Code:* 7001

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the AC is incremented by one in two's complement arithmetic.

*Symbol:* AC + 1 => AC

### ROTATE ACCUMULATOR LEFT (RAL)

*Octal Code:* 7004

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the AC is rotated one binary position to the left with the content of the link. The content of bits AC1 — 11 are shifted to the next greater significant bit, the content of AC0 is shifted into the L, and the content of the L is shifted into AC11.

*Symbol:* ACj => ACj −1
AC0 => L
L => AC11

### ROTATE TWO LEFT (RTL)

*Octal Code:* 7006

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the AC is rotated two binary positions to the left with the content of the link. This instruction is logically equal to two successive RAL operations.

*Symbol:* ACj => ACj −2
AC1 => L
AC0 => AC11
L => AC10

## ROTATE ACCUMULATOR RIGHT (RAR)

*Octal Code:* 7010

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the AC is rotated one binary position to the right with the content of the link. The content of bits AC0 — 10 are shifted to the next less significant bit, the content of AC11 is shifted into the L, and the content of the L is shifted into AC0.

*Symbol:* ACj => ACj +1
AC11 => L
L => AC0

## ROTATE TWO RIGHT (RTR)

*Octal Code:* 7012

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the AC is rotated two binary positions to the right with the content of the link. This instruction is logically equal to two successive RAR operations.

*Symbol:* ACj => ACj +2
AC10 = L
AC11 = AC0
L => AC1

## COMPLEMENT LINK (CML)

*Octal Code:* 7020

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the L is complemented.

*Symbol:* $\overline{L}$ => L

## COMPLEMENT ACCUMULATOR (CMA)

*Octal Code:* 7040

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the AC is set to the one's complement of the current content of the AC. The content of each bit of the AC is complemented individually.

*Symbol:* $\overline{ACj}$ => ACj

## COMPLEMENT AND INCREMENT ACCUMULATOR (CIA)

*Octal Code:* 7041

*Event Time:* 1, 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the AC is converted from a binary value to its equivalent two's complement number. This conversion is accomplished by combining the CMA and IAC commands, thus the content of the AC is complemented during event time 1 and is incremented by one during event time 2.

*Symbol:* $\overline{ACj} \Rightarrow ACj$,
then AC + 1 $\Rightarrow$ AC

## CLEAR LINK (CLL)

*Octal Code:* 7100

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the L is cleared to contain a 0.

*Symbol:* 0 $\Rightarrow$ L

## SET LINK (STL)

*Octal Code:* 7120

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The L is set to contain a binary 1. This instruction is logically equal to combining the CLL and CML commands.

*Symbol:* 1 $\Rightarrow$ L.

## CLEAR ACCUMULATOR (CLA)

*Octal Code:* 7200

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of each bit of the AC is cleared to contain a binary 0.

*Symbol:* 0 $\Rightarrow$ AC

## SET ACCUMULATOR (STA)

*Octal Code:* 7240

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* Each bit of the AC is set to contain a binary 1. This operation is logically equal to combining the CLA and CMA commands.

*Symbol:* 1 = > ACj

Group 2 operate microinstruction format is shown in Figure 6 and the primary microinstructions are explained in the following paragraphs. Any logical combination of bits within this group can be composed into one microinstruction. (The instructions constructed by most logical command combinations are listed in Appendix 1.)

If skips are combined in a single instruction the inclusive OR of the conditions determines the skip when bit 8 is a 0; and the AND of the inverse of the conditions determines the skip when bit 8 is a 1. For example, if ones are designated in bits 6 and 7 (SZA and SNL), the next instruction is skipped if either the content of the AC = 0, or the content of L = 1. If ones are contained in bits 5, 7, and 8, the next instruction is skipped if the AC contains a positive number and the L contains a 0.

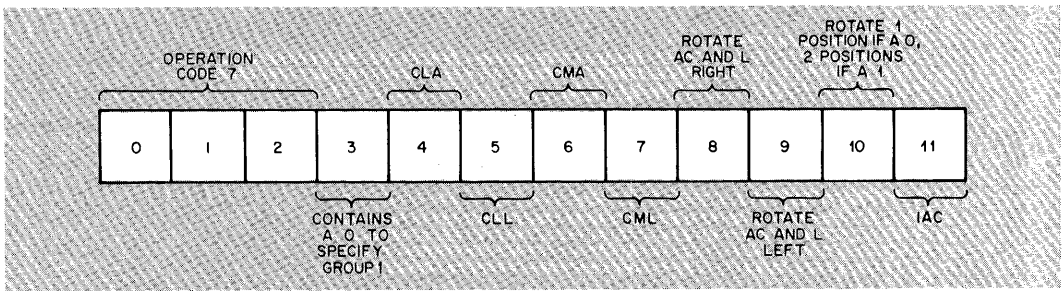



Figure 7 Group 2 Operate Instruction Bit Assignments

## HALT (HLT)

*Octal Code:* 7402

*Event Time:* 1

*Indicators:* OPR, not RUN

*Execution Time:* 1.5 microseconds

*Operation:* Clears the RUN flip-flop at event time 1, so that the program stops at the conclusion of the current machine cycle. This command can be combined with others in the OPR 2 group that are executed during either event time 1, or 2, and so are performed before the program stops.

*Symbol:* 0 = > RUN

## OR WITH SWITCH REGISTER (OSR)

*Octal Code:* 7404

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The inclusive OR operation is performed between the content of the AC and the content of the SR. The result is left in the AC, the original content of the AC is lost, and the content of the SR is unaffected by this command. When combined with the CLA command, the OSR performs a transfer of the content of the SR into the AC.

*Symbol:* ACj V SRj $=>$ ACj

## SKIP, UNCONDITIONAL (SKP)

*Octal Code:* 7410

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* PC + 1 $=>$ PC

## SKIP ON NON-ZERO LINK (SNL)

*Octal Code:* 7420

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the L is sampled, and if it contains a 1 the content of the PC is incremented by one so that the next sequential instruction is skipped. If the L contains a 0, no operation occurs and the next sequential instruction is initiated.

*Symbol:* If L = 1, then PC + 1 $=>$ PC

## SKIP ON ZERO LINK (SZL)

*Octal Code:* 7430

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the L is sampled, and if it contains a 0 the content of the PC is incremented by one so that the next sequential instruction is skipped. If the L contains a 1, no operation occurs and the next sequential instruction is initiated.

*Symbol:* If L = 0, then PC + 1 $=>$ PC

## SKIP ON ZERO ACCUMULATOR (SZA)

*Octal Code:* 7440

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of each bit of the AC is sampled, and if each bit contains a 0 the content of the PC is incremented by one so that the next sequential instruction is skipped. If any bit of the AC contains a 1, no operation occurs and the next sequential instruction is initiated.

*Symbol:* If AC0 − 11 = 0, then PC + 1 => PC

## SKIP ON NON-ZERO ACCUMULATOR (SNA)

*Octal Code:* 7450

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of each bit of the AC is sampled, and if any bit contains a 1 the content of the PC is incremented by one so that the next sequential instruction is skipped. If all bits of the AC contain a 0, no operation occurs and the next sequential instruction is initiated.

*Symbol:* If AC0 − 11 ≠ 0, then PC + 1 => PC

## SKIP ON MINUS ACCUMULATOR (SMA)

*Octal Code:* 7500

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the most significant bit of the AC is sampled, and if it contains a 1, indicating the AC contains a negative two's complement number, the content of the PC is incremented by one so that the next sequential instruction is skipped. If the AC contains a positive number no operation occurs and program control advances to the next sequential instruction.

*Symbol:* If AC0 = 1, then PC + 1 => PC

## SKIP ON POSITIVE ACCUMULATOR (SPA)

*Octal Code:* 7510

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the most significant bit of the AC is sampled, and if it contains a 0, indicating a positive (or zero) two's complement number, the content of the PC is incremented by one so that the next sequential instruction is skipped. If the AC contains a negative number, no operation occurs and program control advances to the next sequential instruction.

*Symbol:* If AC0 = 0, then PC + 1 => PC

### CLEAR ACCUMULATOR (CLA)

*Octal Code:* 7600

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* Each bit of the AC is cleared to contain a binary 0.

*Symbol:* 0 => AC

# CHAPTER 3

# MEMORY AND PROCESSOR BASIC PROGRAMMING

## Memory Addressing

The following terms are used in memory address programming:

| Term | Definition |
|---|---|
| Page | A block of 128 core memory locations ($200_8$ addresses). |
| Current Page | The page containing the instruction being executed; as determined by bits 0 through 4 of the program counter. |
| Page Address | An 8-bit number contained in bits 4 through 11 of an instruction which designates one of 256 core memory locations. Bit 4 of a page address indicates that the location is in the current page when a 1, or indicates it is in page 0 when a 0. Bits 5 through 11 designate one of the 128 locations in the page determined by bit 4. |
| Absolute Address | A 12-bit number used to address any location in core memory. |
| Effective Address | The address of the operand. When the address of the operand is in the current page or in page 0, the effective address is a page address. Otherwise, the effective address is an absolute address stored in the current page or page 0 and obtained by indirect addressing. |

Organization of the standard core memory or any 4096-word field of extended memory is summarized as follows:

| | |
|---|---|
| Total locations (decimal) | 4096 |
| Total addresses (octal) | 7777 |
| Number of pages (decimal) | 32 |
| Page designations (octal) | 0-37 |
| Number of locations per page (decimal) | 128 |
| Addresses within a page (octal) | 0-177 |

Four methods of obtaining the effective address are used as specified by combinations of bits 3 and 4.

| Bit 3 | Bit 4 | Effective Address |
|---|---|---|
| 0 | 0 | The operand is in page 0 at the address specified by bits 5 through 11. |
| 0 | 1 | The operand is in the current page at the address specified by bits 5 through 11. |
| 1 | 0 | The absolute address of the operand is taken from the content of the location in page 0 designated by bits 5 through 11. |
| 1 | 1 | The absolute address of the operand is taken from the content of the location in the current page designated by bits 5 through 11. |

The following example indicates the use of bits 3 and 4 to address any location in core memory. Suppose it is desired to add the content of locations A, B, C, and D to the content of the accumulator by means of a routine stored in page 2. The instructions in this example indicate the operation code, the content of bit 4, the content of bit 3, and a 7-bit address. This routine would take the following form:

| Page 0 Location | Page 0 Content | Page 1 Location | Page 1 Content | Page 2 Location | Page 2 Content | Remarks |
|---|---|---|---|---|---|---|
| | | | | R | TAD 00 A | DIRECT TO DATA IN PAGE 0 |
| | | | | S | TAD 01 B | DIRECT TO DATA IN SAME PAGE |
| | | | | T | TAD 10 M | INDIRECT TO ADDRESS SPECIFIED IN PAGE 0 |
| | | | | U | TAD 11 N | INDIRECT TO ADDRESS SPECIFIED IN SAME PAGE |
| | | | | | . | |
| | | | | | . | |
| | | | | | . | |
| | | | | | . | |
| A | xxxx | C | xxxx | B | xxxx | |
| M | C | D | xxxx | N | D | |

Routines using 128 instructions, or less, can be written in one page using direct addresses for looping and using indirect addresses for data stored in other pages. When planning the location of instructions and data in core memory, remember that the following locations are reserved for special purposes:

| Address | Purpose |
|---|---|
| $0_8$ | Stores the contents of the program counter following a program interrupt. |
| $1_8$ | Stores the first instruction to be executed following a program interrupt. |
| $10_8$ through $17_8$ | Auto-indexing. |

## INDIRECT ADDRESSING

When indirect addressing is specified, the address part (bits 5-11) of a memory reference instruction is interpreted as the address of a location containing not the operand, but containing the address of the operand. Consider the instruction TAD A. Normally, A is interpreted as the address of the location containing the quantity to be added to the content of the AC. Thus, if location 100 contains the number 5432, the instruction TAD 100 causes the quantity 5432 to be added to the content of the AC. Now suppose that location 5432 contains the number 6543. The instruction TAD I 100 (where I signifies indirect addressing) causes the computer to take the number 5432, which is in location 100, as the effective address of the instruction and the number in location 5432 as the operand. Hence, this instruction results in the quantity 6543 being added to the content of the AC.

## AUTO-INDEXING

When a location between $10_8$ and $17_8$ in page 0 of any core memroy field is addressed indirectly (by an instruction in which bit 3 is a 1) the content of that location is read, incremented by one, rewritten in the same location, and then taken as the effective address of the instruction. This feature is called auto-indexing. If location $12_8$ contains the number 5432 and the instruction DCA I Z 12 is given, the number 5433 is stored in location 12, and the content of the accumulator is deposited in core memory location 5433.

# Storing and Loading

Data is stored in any core memory location by use of the DCA Y instruction. This instruction clears the AC to simplify loading of the next datum. If the data deposited is required in the AC for the next program operation, the DCA must be followed by a TAD Y for the same address.

All loading of core memory information into the AC is accomplished by means of the TAD Y instruction, preceded by an instruction that clears the AC such as CLA or DCA.

Storing and loading of information in sequential core memory locations can make excellent use of an auto-index register to specify the core memory address.

# Program Control

Transfer of program control to any core memory location uses the JMP or JMS instructions. The JMP I (indirect address, 1 in bit 3) is used to transfer program control to any location in core memory which is not in the current page or page 0.

The JMS Y is used to enter a subroutine which starts at location Y +1 in the current page or page 0. The content of the PC + 1 is stored in the specified address Y, and address Y + 1 is transferred into the PC. To exit a subroutine the last instruction is a JMP I Y, which returns program control to the location stored in Y.

# Indexing Operations

External events can be counted by the program and the number can be stored in core memory. The core memory location used to store the event count can be initialized (cleared) by a CLA command followed by a DCA instruction. Each time the event occurs, the event count can be advanced by a sequence of commands such as CLA, TAD, IAC, and DCA.

The ISZ instruction is used to count repetitive program operations or external events without disturbing the content of the accumulator. Counting a specified number of operations is performed by storing a two's complement negative number equal to the number of iterations to be counted. Each time the operation is performed, the ISZ instruction is used to increment the content of this stored number and check the result. When the stored number becomes zero, the specified number of operations have occured and the program skips out of the loop and back to the main sequence.

This instruction is also used for other routines in which the content of a memory location is incremented without disturbing the content of the accumulator, such as storing information from an I/O device in sequential memory locations or using core memory locations to count I/O device events.

# Logic Operations

The PDP-8 instruction list includes the logic instruction, AND Y. From this instruction short routines can be written to perform the inclusive OR and exclusive OR operations.

## LOGICAL AND

The logic AND operation between the content of the accumulator and the content of a core memory location Y is performed directly by means of the AND Y instruction. The result remains in the AC, the original content of the AC is lost, and the content of Y is unaffected.

## INCLUSIVE OR

Assuming value A is in the AC and value B is stored in a known core memory address, the following sequence performs the inclusive OR. The sequence is stated as a utility subroutine called IOR.

```
/CALLING SEQUENCE                              JMS IOR
/                                              (ADDRESS OF B)
/                                              (RETURN)
/ENTER WITH ARGUMENT IN AC; EXIT WITH LOGICAL RESULT IN AC

          IOR,    0
                  DCA TEM1
                  TAD I IOR
                  DCA TEM2
                  TAD TEM1
                  CMA
                  AND I TEM2
                  TAD TEM1
                  ISZ IOR
                  JMP I IOR
          TEM1,   0
          TEM2,   0
```

## EXCLUSIVE OR

The exclusive OR operation for two numbers, A and B, can be performed by a subroutine called by the mnemonic code XOR. In the following general purpose XOR subroutine, the value A is assumed to be in the AC, and the address of the value B is assumed to be stored in a known core memory location.

```
/CALLING SEQUENCE                                   JMS XOR
/                                                   (ADDRESS OF B)
/                                                   (RETURN)
/ENTER WITH ARGUMENT IN AC; EXIT WITH LOGICAL RESULT IN AC

XOR,    0
        DCA TEM1
        TAD I XOR
        DCA TEM2
        TAD TEM1
        AND I TEM2
        CMA IAC
        CLL RAL
        TAD TEM1
        TAD I TEM2
        ISZ XOR
        JMP I XOR
TEM1,   0
TEM2,   0
```

An XOR subroutine can be written using fewer core memory locations by making use of the IOR subroutine; however, such a subroutine takes more time to execute. A faster XOR subroutine can be written by storing the value B in the second instruction of the calling sequence instead of the address of B; however, the resulting subroutine is not as utilitarian as the routine given here.

# Arithmetic Operations

One arithmetic instruction is included in the PDP-8 order code, the two's complement add: TAD Y. Using this instruction, routines can easily be written to perform addition, subtraction, multiplication, and division in two's complement arithmetic.

## TWO'S COMPLEMENT ARITHMETIC

In two's complement arithmetic addition, subtraction, multiplication, and division of binary numbers is performed in accordance with the common rules of binary arithmetic. In PDP-8, as in other machines utilizing complementation techniques, negative numbers are represented as the complement of positive numbers, and subtraction is achieved by complement addition. Representation of negative values in one's complement arithmetic is slightly different from that in two's complement arithmetic.

The one's complement of a number is the complement of the absolute positive value; that is, all ones are replaced by zeros and all zeros are replaced by ones. The two's complement of a number is equal to the one's complement of the positive value plus one.

In one's complement arithmetic a carry from the sign bit (most significant bit) is added to the least significant bit in an end-around carry. In two's complement arithmetic a carry from the sign bit complements the link (a carry would set the link to 1 if it were properly cleared before the operation), and there is no end-around carry.

A one's complement representation of a negative number is always one less than the two's complement representation of the same number. Differences between one's and two's complement representations are indicated in the following list.

| Number | 1's Complement | 2's Complement |
|---|---|---|
| +5 | 000000000101 | 000000000101 |
| +4 | 000000000100 | 000000000100 |
| +3 | 000000000011 | 000000000011 |
| +2 | 000000000010 | 000000000010 |
| +1 | 000000000001 | 000000000001 |
| +0 | 000000000000 | 000000000000 |
| −0 | 111111111111 | Nonexistent |
| −1 | 111111111110 | 111111111111 |
| −2 | 111111111101 | 111111111110 |
| −3 | 111111111100 | 111111111101 |
| −4 | 111111111011 | 111111111100 |
| −5 | 111111111010 | 111111111011 |

Note that in two's complement there is only one representation for the number which has the value zero, while in one's complement there are two representations. Note also that complementation does not interfere with sign notation in either one's complement or two's complement arithmetic; bit 0 remains a 0 for positive numbers and a 1 for negative numbers.

To form the two's complement of any number in the PDP-8, the one's complement is formed, and the result is incremented by one. This is accomplished by the instruction CMA combined with an IAC instruction. Since both of these instructions are functions of the OPR 1 instruction and the actions occur at different event times, they can be combined to form the instruction CIA, Complement and Increment AC.

## ADDITION

The addition of a number contained in a core memory location and the number contained in the accumulator is performed directly by using the TAD Y instruction, assuming that the binary point is in the same position and that both numbers are properly represented in two's complement arithmetic. Addition can be performed without regard for the sign of either the augend or the addend. Overflow is possible, in which case the result will have an incorrect sign, although the 11 least significant bits will be correct. Following the addition a test for overflow can be made by using the SZL command.

## SUBTRACTION

Subtraction is performed by complementing the subtrahend and adding the minuend. As in addition, if both numbers are represented by their two's complement, subtraction can be performed without regard for the sign of either number. Assuming that both numbers are stored in core memory, a routine to find the value of A-B follows:

```
CLA
TAD B          /LOAD SUBTRAHEND INTO AC
CIA            /COMPLEMENT AND INCREMENT B
TAD A          /AC = A − B
```

# CHAPTER 4

# PROGRAM INTERRUPT

The program interrupt feature allows certain external conditions to interrupt the computer program. It is used to speed the information processing of input-output devices or to allow certain alarms to halt the program in progress and initiate another routine. When a program interrupt request is made the computer completes execution of the instruction in progress before acknowledging the request and entering the interrupt mode. A program interrupt is similar to a JMS to location 0; that is, the content of the program counter is stored in location 0, and the program resumes operation in location 1. The interrupt program commencing in location 1 is responsible for identifying the signal causing the interruption, for removing the interrupt condition, and for returning to the original program. Exit from the interrupt program, back to the original program, can be accomplished by a JMP I Z 0 instruction.

## Instructions

The two instructions associated with the program interrupt synchronization element are IOT microinstructions that do not use the IOP generator. These instructions are:

INTERRUPT TURN ON (ION)

*Octal Code:* 6001

*Event Time:* Not applicable

*Indicators:* IOT, FETCH, ION

*Execution Time:* 1.5 microseconds

*Operation:* This command enables the computer to respond to a program interrupt request. If the interrupt is disabled when this instruction is given, the computer executes the next instruction, then enables the interrupt. The additional instruction allows exit from the interrupt subroutine before allowing another interrupt to occur. This instruction has no affect upon the condition of the interrupt circuits if it is given when the interrupt is enabled.

*Symbol:* 1 => INT. ENABLE

INTERRUPT TURN OFF (IOF)

*Octal Code:* 6002

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* This command disables the program interrupt synchronization element to prevent interruption of the current program.

*Symbol:* 0 => INT. ENABLE, INT. DELAY

## Programming

When an interrupt request is acknowledged, the interrupt is automatically disabled by the program interrupt synchronization circuits (not by instructions). The next instruc-

tion is taken from core memory location 1. Usually the instruction stored in location 1 is a JMP, which transfers program control to a subroutine which services the interrupt. At some time during this subroutine an ION instruction must be given. The ION can be given at the end of the subroutine to allow other interrupts to be serviced after program control is transferred back to the original program. In this application, the ION instruction immediately precedes the last instruction in the routine. A delay of one instruction (regardless of the execution time of the following instruction) is inherent in the ION instruction to allow transfer of program control back to the original program before enabling the interrupt. Usually exit from the subroutine is accomplished by a JMP I Z 0 instruction.

The ION command can be given during the subroutine as soon as it has determined the I/O device causing the interrupt. This latter method allows the subroutine which is handling a low priority interrupt to be interrupted, possibly by a high priority device. Programming of an interrupt subroutine which checks for priority and allows itself to be interrupted, must make provisions to relocate the content of the program counter stored in location 0; so that if interrupted, the content of the PC during the subroutine is stored in location 0, and the content of the PC during the original program is not lost.

# CHAPTER 5

# DATA BREAK

Peripheral equipment connected to the data break facility can cause a temporary suspension in the program in progress to transfer information with the computer core memory, via the MB. One I/O device can be connected directly to the data break facility or up to seven devices can be connected to it through the Type DM01 Data Multiplexer. This cycle stealing mode of operation provides a high-speed transfer of individual words or blocks of information at core memory addresses specified by the I/O device. Since program execution is not involved in these transfers, the program counter, accumulator, and instruction register are not disturbed or involved in these transfers. The program is merely suspended at the conclusion of an instruction execution and the data break is entered to perform the transfer, then the Fetch state is entered to continue the main program.

Data breaks are of two basic types: single-cycle and three-cycle. In a single-cycle data break, registers in the device (or device interface) specify the core memory address of each transfer and count the number of transfers to determine the end of data blocks. In the three-cycle data break two computer core memory locations perform these functions, simplifying the device interface by omitting two hardware registers.

The computer receives the following signals from the device during a data break:

| Signal | −3 Volts | 0 Volts |
|---|---|---|
| Break Request | No break request | Break request |
| Cycle Select | One-cycle break | Three-cycle break |
| Transfer Direction | Data into PDP-8 | Data out of PDP-8 |
| Increment CA Inhibit | CA incremented | CA not incremented |
| Increment MB (pulse) | MB not incremented | MB incremented |
| Address (12 bits) | Binary 0 | Binary 1 |
| Data (12 bits) | Binary 0 | Binary 1 |

The computer sends the following signals to the device during a data break:

| Signal | Characteristics |
|---|---|
| Data (12 bits) | −3 volts = binary 0, 0 volts = binary 1 |
| Address Accepted | 400-nanosecond negative pulse beginning at memory done time |
| WC Overflow | 400-nanosecond negative pulse occurring at T1 time |
| Buffered Break | −3 volts when in Break state |

To initiate a data break an I/O device must supply four signals simultaneously to the data break facility. These signals are the Break Request signal, which sets the BRK SYNC flip-flop in the major state generator to control entry into the data break states (Word Count for a three-cycle data break or Break for a single-cycle data break); a Transfer Direction signal, supplied to the MB control element to allow data to be strobed into the MB from the peripheral equipment and to inhibit reading from core memory; a Cycle Select signal which controls gating in the major state generator to determine if the one-cycle or three-cycle data break is to be selected; and a core memory address of the transfer which is supplied to the input of the MA. When the

break request is made, the data break replaces entry into the Fetch state of an instruction. Therefore the data break is entered at the conclusion of the Execute state of most memory reference instructions and at the conclusion of a Fetch state of augmented instructions. Having established the data break, each machine cycle is a Word Count, Current Address, or Break cycle until all data transfers have taken place, as indicated by removal of the Break Request signal by the peripheral equipment.

More exactly, the Break Request signal enables a diode-capacitor-diode gate at the binary 1 input of the BRK SYNC flip-flop. Midway through each computer cycle (T1) this gate is pulsed to set the flip-flop if the Break Request signal has been received.

At the beginning (T2) of each machine cycle the major state generator is set to establish the state for the cycle. At this time the status of the BRK SYNC flip-flop is sampled and the flip-flop is cleared. If the BRK SYNC flip-flop is in the 1 state at this time, the Word Count or Break state is set into the major state generator and a data break commences.

Therefore, to initiate a data break, the Break Request must be at ground potential for at least 400 nanoseconds preceding T1 of the cycle preceding the data break cycle. A Break Request signal should be supplied to the computer when the address, data, Transfer Direction and Cycle Select signals are supplied to the computer, and not before.

When a data break occurs, the address designated by the device is loaded into the MA during time T2 of the last cycle of the current instruction, and the major state generator is set to the Word Count state if the Cycle Select signal is at ground, or is set to the Break state if this signal is at −3 volts. The program is delayed for the duration of the data break, commencing in the following cycle. A break request is granted only after completion of the current instruction as specified by the following conditions:

1. At the end of the Fetch cycle of an OPR or IOT instruction, or a directly-addressed JMP instruction.

2. At the end of the Defer cycle of an indirectly addressed JMP instruction.

3. At the end of the Execute cycle of a JMS, DCA, ISZ, TAD, or AND instruction.

At the beginning of the Word Count cycle of a three-cycle data break or the Break cycle of a one-cycle data break the address supplied to the input of the MA is strobed into the MA and the computer supplies an Address Accepted pulse to the device. Entry into the Break cycle is indicated to the peripheral equipment by a Buffered Break signal and by an Address Accepted pulse that can be used to enable gates in the device to perform tasks associated with the transfers. The Addresss Accepted pulse is the most convenient control to be used by I/O equipment to disable the Break Request signal, since this signal must be removed at the end of T2 time to prevent continuance at the data break into the next cycle. Also at the beginning of the Break cycle, the MB is cleared in preparation for receipt of data from either the core memory or the external device. If the Transfer Direction signal establishes the direction as out of the computer, the content of the core memory register at the address specified is transferred into the MB and is immediately available for strobing by the peripheral equipment. If the Transfer Direction signal specifies a data direction into the PDP-8, reading from core memory is inhibited and data is transferred into the MB from peripheral equipment.

The status of the BRK SYNC flip-flop is sensed at the beginning of a Break cycle to determine if an additional Break cycle is required. If a Break Request signal has been received since T2, the Break state is maintained in the major state generator; if the Break Request signal has not been received by this time; the Fetch state is set into the major state generator to continue the program. The Break Request signal should be removed by the end of the Address Accepted signal if additional Break cycles are not required.

## Single-Cycle Data Break

One-cycle breaks transfer a data word into the computer core memory from the device, transfer a data word into a device from the core memory, or increment the content of a device-specified core memory location. In each of these types of data break one computer cycle is stolen from the program by each transfer; Break cycles occur singly (interleaved with the program steps) or continuously (as in a block transfer), depending upon the timing of the Break Request signal.

During the memory strobe portion of the Break cycle, the content of the addressed cell is read into the MB if the transfer direction is out of the computer (into the I/O device). If the transfer direction is into the computer, generation of the Memory Strobe pulse is inhibited so that the MB (cleared during the previous cycle) remains cleared. Information is transferred from the output data register of the I/O device into the MB and is written into core memory during time T1 of the Break cycle. In an outward transfer, the write operation restores the original content of the address cell to memory.

The MB is cleared during time T2 of the Break cycle. If there is a further break request, another Break cycle is initiated. If there is no break request, the content of the PC is transferred into the MA, the IR is cleared, and the major state generator is set to Fetch. The program then executes the next instruction.

The increment MB facility is useful for counting iterations or events by means of a data break, so that the PC and AC are not disturbed. Within one Break cycle of 1.5 microseconds, a word is fetched from a device-specified core memory location, is incremented by one, and is restored to the same memory location. The Increment MB signal input must be supplied to the computer only during a Break cycle in which the direction of transfer is out of the PDP-8. These restrictions can be met by a simple AND gate in the device; an Increment MB signal is generated only when an event occurs, the Buffered Break signal from the computer is present, and the Transfer Direction signal supplied to the computer is at ground potential.

## Three-Cycle Data Break

The three-cycle data break provides an economical method of controlling the transfer of data between the computer core memory and fast peripheral devices. Transfer rates in excess of 220 kc are possible using this feature of the PDP-8.

The three-cycle data break differs from the one-cycle break in that a ground-level Cycle Select signal is supplied so that when the data break conditions are fulfilled the program is suspended and the Word Count state is entered. The Word Count state is entered to increment the fixed core memory location containing the word count. The device requesting the break supplies this address as in the one-cycle break, except that this is a fixed address supplied by wired ground and −3v signals rather than from a register. The only restriction on this address is that it must be an even number (bit 11 = 0).

Following the Word Count state a Current Address state occurs in which the location following the Word Count address (bit 11 = 1 after + 1 ⇒ MA) is read, incremented by one, restored to memory, and loaded into the MA to be used as the transfer address. Then the normal Break state is entered to effect the transfer between the device and the computer memory cell specified by the MA.

## WORD COUNT STATE

When this state is entered the content of the core memory address specified by the external device is read into the MB during time state T1. The word count, established previously by instructions, is the 2's complement negative number equal to the required number of transfers. The word in the MB is incremented by 1 to advance the word count, and if the word becomes 0 when incremented, the computer generates a WC Overflow pulse and supplies it to the device. During time T2 the incremented word count is rewritten in memory, the MB is cleared, the content of the MA is incremented by 1 to establish the next location as the address for the following memory cycle, and the major state generator is set to the Current Address state.

## CURRENT ADDRESS STATE

Operations during the second cycle of the three-cycle data break depend upon the condition of the Increment CA Inhibit ( 1 → CA Inhibit) signal supplied to the computer from the I/O device. During T1 the address following the word count is read into the MB. If the Increment CA Inhibit signal is at ground potential, no further operations occur during T1. If this signal is at −3v, the content of the MB is incremented by 1 during T1 to advance the address of the transfer to the next sequential location. During T2, the content of the MB is rewritten into core memory, the address word in the MB is transferred into the MA to designate the address to be used in the succeeding memory cycle, the MB is cleared, and the major state generator is set to the Break state.

## BREAK STATE

The actual transfer of data between the external device and the core memory, through the MB, occurs during the Break state as during a single-cycle data break, except that the address is determined by the current content of the MA rather than directly by the device.

# CHAPTER 6

# MEMORY EXTENSION CONTROL TYPE 183 AND MEMORY MODULE TYPE 184

Extension of the storage capacity of the standard 4096-word core memory is accomplished by adding fields of 4096-word core memories, each field being a Type 184 Memory Module. Field select control and address extension control for Type 184 Memory Modules are provided by the Type 183 Memory Extension Control. Up to seven fields can be added to the standard 4096-word memory, providing a maximum storage of 32,768 words. Direct addressing of 32,768 words require 15 binary bits ($2^{15} = 32{,}768$). However, since programs and data need not be directly addressed for execution of each instruction, a field can be program-selected, and all 12-bit addresses are then assumed to be within the current memory field. Program interrupt of a program in any field automatically specifies field 0, address 0 for storage of the program count. The memory extension control consists of several 3-bit flip-flop registers that extend addresses to 15 bits to establish or select a field.

Addition of a memory extension control to a standard PDP-8 requires a simple modification of the operator console to activate indicators and switches associated with the instruction field register and the data field register of the control. These switches function in the same manner as the switch register, to load information into associated registers when the LOAD ADDRESS key is pressed.

The seven functional circuit elements which comprise the memory extension control perform as follows:

*Instruction Field Register (IF):* The IF is a 3-bit register that serves as an extension of the PC. The content of the IF determines the field from which all instructions are taken and the field from which operands are taken in directly-addressed AND, TAD, ISZ, or DCA instructions. Operating the LOAD ADDRESS key clears the IF, then sets it by a transfer of ones from the INSTRUCTION FIELD switch register on the operator console. During a JMP or JMS instruction the IF is set by a transfer of information contained in the instruction buffer register. When a program interrupt occurs, the content of the IF is automatically stored in bits 0 through 2 of the save field register for restoration to the IF from the instruction buffer register at the conclusion of the program interrupt subroutine.

*Data Field Register (DF):* This 3-bit register determines the memory field from which operands are taken in indirectly-addressed AND, TAD, ISZ, or DCA instructions. The DF is cleared and set by a ones transfer of information contained in the DATA FIELD switch register by operation of the LOAD ADDRESS key. The DF is set by a transfer of information from bits 6 through 8 of the MB during a CDF microinstruction to establish a microprogrammed data field. When a program interrupt occurs, the content of the DF is automatically stored in the save field register. The DF is set by a transfer of information from bits 3 through 5 of the save field register by the RMF microinstruction to restore the data field at the conclusion of the program interrupt subroutine.

*Instruction Buffer Register (IB):* The IB serves as a 3- bit input buffer for the instruction field register. All field number transfers into the instruction field register are made through the instruction buffer, except transfers from the operator console switches. The IB is cleared and set by operation of the LOAD ADDRESS key in the same manner as the instruction field register. A CIF microinstruction loads the IB with the programmed field number contained in MB 6-8. An RMF microinstruction transfers the content of bits 0 through 2 of the save field register into the IB to restore the instruction field to the conditions that existed prior to a program interrupt.

*Save Field Register (SF):* When a program interrupt occurs, this 6-bit register is cleared, then loaded from the instruction field and data field registers. The RMF microinstruction can be given immediately prior to the exit from the program interrupt subroutine to restore the instruction field and data field by transferring the content of the SF into the instruction buffer and the data field register. The SF is cleared during the cycle in which the program count is stored at address 0000 of the JMS instruction forced by a program interrupt request, then the instruction field and data field are strobed into the SF.

*Start Field Signal Generator:* When the PDP-8 core memory capacity is extended, the standard memory is designated as field 0. This circuit produces the Enable Field 0 signal when data field 0 is selected, instruction field 0 is selected, or when break field 0 is selected. Similar circuits are provided for each of the other (up to seven) fields.

*Accumulator Transfer Gating:* This gating allows the content of the save field register, instruction field register, or the data field register to be strobed into the accumulator. Transfer of information in this manner is accomplished by circuits which sample the content of registers and supply positive pulses to the AC upon receipt of IOT command pulses. During an RIB microinstruction, bits 6 through 11 of the AC are set by the content of the save field register. During an RIF microinstruction, bits 6 through 8 of the AC are set by the content of the instruction field register. During an RDF microinstruction, bits 6 through 8 of the AC are set by the content of the data field register.

*Device Selector:* Bits 3 through 5 of the IOT instruction are decoded to produce the IOT command pulses for the memory extension control. Bits 6 through 8 of the instruction are not used for device selection since they specify a field number in some commands. Therefore, the select code for this device selector is designated as 2X.

Each Type 184 Memory Module consists of a core array, address selection circuits, inhibit selection circuits, sense amplifiers, and memory drivers which are identical with these in the standard PDP-8.

# Instructions

The instructions for the Type 183 option do not use the IOP generator and extend the IOT instruction list to include the following:

### CHANGE TO DATA FIELD N (CDF)

*Octal Code:* 62N1

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The data field register is loaded with the program-selected field number (N = 0 to 7). All subsequent memory requests for operands are automatically switched to that data field until the data field number is changed by a new CDF command, or during a program interrupt.

*Symbol:* MB6 − 8 = > DF

### CHANGE INSTRUCTION FIELD (CIF)

*Octal Code:* 62N2

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The instruction buffer register is loaded with the program-selected field number (N = 0 to 7). The next JMP or JMS instruction causes the new field to be entered.

*Symbol:* MB6 − 8 = > IB

### READ DATA FIELD (RDF)

*Octal Code:* 6214

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the data field register is transferred into bits 6, 7, 8 of the AC. All other bits of the AC are unaffected.

*Symbol:* DF = > AC6 − 8

### READ INSTRUCTION FIELD (RIF)

*Octal Code:* 6224

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the instruction field register is transferred into bits 6, 7, 8 of the AC. All other bits of the AC are unaffected.

*Symbol:* IF = > AC6 − 8

READ INTERRUPT BUFFER (RIB)

*Octal Code:* 6234

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The instruction field and data field held in the save field register during a program interrupt are transferred into bits 6 through 8, and 9 through 11 of the AC respectively.

*Symbol:* SF 0 – 2 => AC6 – 8
SF 3 – 5 => AC9 – 11

RESTORE MEMORY FIELD (RMF)

*Octal Code:* 6244

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* This command is used upon exit from the program interrupt subroutine in another field. The data and instruction fields that were interrupted by the subroutine are restored by transferring the content of the save field register into the instruction buffer and data field registers.

*Symbol:* SF 0 – 2 => IB
SF 3 – 5 => DF

# Programming

Instructions and data are accessed from the currently assigned instruction and data fields, where instructions and data may be stored in the same or different memory fields. When indirect memory references are executed, the operand address refers first to the instruction field to obtain an effective address, which in turn, refers to a location in the currently assigned data field. All instructions and operands are obtained from the field designated by the content of the instruction field register, except for indirectly-addressed operands which are specified by the content of the data field register. In other words, the DF is effective only in the Execute cycle that is directly preceded by the Defer cycle of a memory reference instructions, as follows:

| Indirect (Bit 3) | Page or Z Bit (Bit 0) | Field In IF | Field In DF | Effective Address |
|---|---|---|---|---|
| 0 | 0 | m | n | The operand is in page 0 of field m at the page address specified by bits 5 through 11. |
| 0 | 1 | m | n | The operand is in the current page of field m at the page address specified by bits 5 through 11. |
| 1 | 0 | m | n | The absolute address of the operand in field n is taken from the content of the location in page 0 of field m designated by bits 5 through 11. |
| 1 | 1 | m | n | The absolute address of the operand in field n is taken from the content of the location in the current page of field m designated by bits 5 through 11. |

Each field of extended memory contains eight autoindex registers in addresses 10 through 17. For example, assume that a program in field 2 is running (IF = 2) and using operands in field 1 (DF = 1) when the instruction TAD I 10 is fetched. The Defer cycle is entered (bit 3 = 1) and the content of location 10 in field 2 is read, incremented, and rewritten. If address 10 in field 2 originally contained 4321, it now contains 4322. In the Execute cycle the operand is fetched from location 4322 of field 1.

Program control is transferred between memory fields by the CIF commands. This instruction does not change the instruction field directly, since this would make it impossible to execute the next sequential instruction. The CIF instruction sets the new instruction field into the IB for automatic transfer into the IF when either a JMP or JMS instruction is executed. The DF is unaffected by the JMP and JMS instructions. The 12-bit program counter is set in the normal manner and, since the IF is an extension on the most significant end of the PC, program sequence resumes in the new memory field following a JMP or JMS. Entry into a program interrupt is inhibited after the CIF instruction until a JMP or JMS is executed.

To call a subroutine that is out of the current field, the data field register is set to indicate the field of the calling JMS, which establishes the location of the operands as well as the identity of the return field. The instruction field is set to the field of the starting address of the subroutine. The following sequence returns program control to the main program from a subroutine that is out of the current field.

```
/PROGRAM OPERATIONS IN MEMORY FIELD 2
/INSTRUCTION FIELD = 2; DATA FIELD = 2
/CALL A SUBROUTINE IN MEMORY FIELD 1
/INDICATE CALLING FIELD LOCATION BY THE CONTENT OF THE DATA FIELD

            CIF         10          /CHANGE TO INSTRUCTION
                                    /FIELD 1 = 6212
            JMS         1  SUBRP    /SUBRP = ENTRY ADDRESS
            CDF         20          /RESTORE DATA FIELD
SUBRP,      SUBR                    /POINTER
/CALLED SUBROUTINE
            0                       /SUBR = PC + 1 AT CALLING POINT
            RDF                     /READ DATA FIELD INTO AC
            TAD  RETURN             /CONTENT OF THE AC = 6202 + DATA
                                    /FIELD BITS
            DCA  EXIT               /STORE INSTRUCTION SUBROUTINE
              .
              .
              .
              .
EXIT,       0                       /A CIF INSTRUCTION
            JMP I SUBR              /RETURN
RETURN,     CIF
```

When a program interrupt occurs, the current instruction and data field numbers are automatically stored in the 6-bit save field register, then the IF and DF are cleared. The 12-bit program count is stored in location 0000 of field 0 and program control advances to location 0001 of field 0. At the end of the program interrupt subroutine the RMF instruction restores the IF and DF from the content of the SF. The following instruction sequence at the end of the program interrupt subroutine continues the interrupted program after the interrupt has been processed:

```
        .           /RESTORE MQ IF REQUIRED
        .
        .
        .
        .           /RESTORE L IF REQUIRED
        .
        .
        .
CLA
TAD   AC            /RESTORE AC
RMF                 /LOAD IB FROM SF
ION                 /TURN ON INTERRUPT SYSTEM
JMP   I 0           /RESTORE PC WITH CONTENT OF
                    /LOCATION 0 AND LOAD IF FROM IB
```

A device using the computer data break facility supplies a 12-bit address to the MA and a 3-bit address extension to the Memory Extension Control Type 183. The address extension is received by a break field decoder which selects the memory field used for the data break.

# CHAPTER 7

# MEMORY PARITY TYPE 188

Data transmission checking of each word written in and read from core memory is provided by this option. The option replaces the 12-bit core memory with a 13-bit system (driving, inhibiting, sensing circuits as well as a core array constructed of 13 planes) and includes a parity generator and a parity checking circuit. The parity generator produces the 13th bit for each 12-bit data word written in core memory so that the entire word contains an odd number of binary ones. The parity checking circuit monitors each word read from core memory to assure that the odd parity is maintained. If a word read contains an even number of ones a transmission error is indicated by setting a parity error flag. This flag is connected to the program interrupt synchronization element of the computer to initiate a program interrupt subroutine. This routine sequentially checks all equipment error flags to determine the option causing the interrupt and initiates an appropriate service and returns to the main program, or provides a suitable error printout and halts programmed operations. Upon determining that a memory parity error has occurred the program interrupt subroutine can repeat the main program step that caused the error to check the reliability of the error condition, can perform a simple write/read/check routine at the error address, or can determine the status of the machine when the error was detected and re-establish or print out these conditions and halt.

## Instructions

Two instructions are associated with the Type 188 option. They are:

SKIP ON NO MEMORY PARITY ERROR (SMP)

*Octal Code:* 6104

*Event Time:* 3

*Indicator:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The memory parity error flag is sensed and if it contains a 0 (signifying no error has been detected) the PC is incremented so that the next successive instruction is skipped.
*Symbol:* If Memory Parity Error Flag = 0, then PC + 1 => PC

CLEAR MEMORY PARITY ERROR FLAG (CMP)

*Octal Code:* 6102

*Event Time:* 2

*Indicator:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The memory parity error flag is cleared.

*Symbol:* 0 => Memory Parity Error Flag

# Programming

Both instructions for this option are used in the program interrupt subroutine and in diagnostic maintenance programs. The SMP command is used as a programmed check for memory parity error. In the program interrupt subroutine this command can be followed by a jump to a portion of the routine that services the memory parity option as described previously. The CMP command is used to initialize the memory parity option in preparation for normal programmed operation of the computer.

# CHAPTER 8

# EXTENDED ARITHMETIC ELEMENT TYPE 182

This option consists of circuits that perform parallel arithmetic operations on positive binary numbers. A 12-bit multiplier quotient register (MQ), a 5-stage step counter (SC), and various shifting and control logic constitute the option. The AC and MB are used in conjunction with these logic elements to perform arithmetic operations. With the addition of this option to a PDP-8 system, indicators on the operator console for the content of each bit of the MQ are activated and a class of instructions is added to the Group 2 Operate instruction list.

## Instructions

The exended arithmetic element (EAE) microinstructions are specified by an operate instruction (operation code 7) in which bits 3 and 11 contain binary ones. Being augmented instructions, the EAE commands are microprogrammed and can be combined with each other to perform non-conflicting logical operations. Format and bit assignments of the EAE commands are indicated in Figure 8.

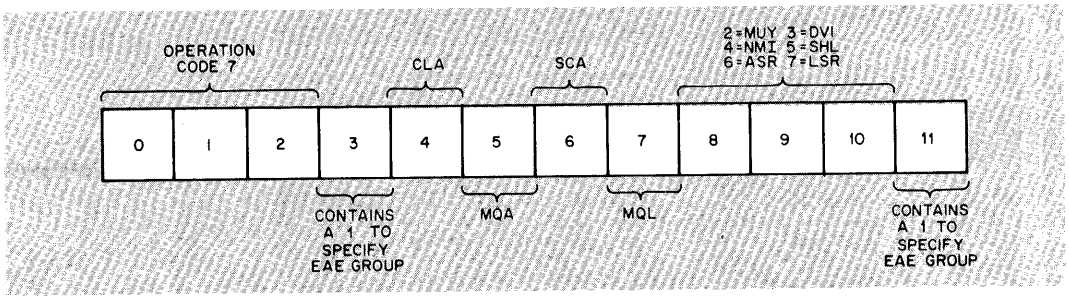



Figure 8 EAE Microinstruction Bit Assignments

### MULTIPLY (MUY)

*Octal Code:* 7405

*Event Time:* 2

*Indicators:* OPR, FETCH, PAUSE

*Execution Time:* 9.0 to 21.0 microseconds

*Operation:* The number held in the MQ is multiplied by the number held in core memory location PC + 1 (or the next successive core memory location after the MUY command). At the conclusion of this command the link contains a 0, the most significant 12 bits of the product are contained in the AC and the least significant 12 bits of the product are contained in the MQ.

*Symbol:* Y $\times$ MQ $=>$ AC, MQ
0 $=>$ L

## DIVIDE (DVI)

*Octal Code:* 7407

*Event Time:* 2

*Indicators:* OPR, FETCH, PAUSE

*Execution Time:* 36.5 microseconds or less

*Operation:* The 24-bit dividend held in the AC (most significant 12 bits) and the MQ (least significant 12 bits) is divided by the divisor held in core memory location PC + 1 (or the next successive core memory location following the DVI command). At the conclusion of this command the quotient is held in the MQ, the remainder is in the AC, and the L contains a 0. If the L contains a 1, divide overflow occurred so the operation was concluded after the first cycle of the division.

*Symbol:* AC, MQ ÷ Y => MQ

## NORMALIZE (NMI)

*Octal Code:* 7411

*Event Time:* 2

*Indicators:* OPR, FETCH, PAUSE

*Execution Time:* 1.5 microseconds + 0.5 microsecond for each shift

*Operation:* This instruction is used as part of the conversion of a binary number to a fraction and an exponent for use in floating-point arithmetic. The combined content of the AC and the MQ is shifted left by this one command until the content of AC0 is not equal to the content of AC1, or until 6000 0000 is contained in the combined AC and MQ, to form the fraction. Zeros are shifted into vacated MQ11 positions for each shift. At the conclusion of this operation, the step counter contains a number equal to the number of shifts performed, which can be loaded into the AC by an SCA command to form the exponent. The content of L is lost. Both positive and negative two's complement numbers can be normalized.

*Symbol:* ACj => ACj − 1
AC0 => L
MQ0 => AC11
MQj = MQj − 1
0 => MQ11 until AC0 ≠ AC1 or until AC MQ = 6000 0000

## SHIFT ARITHMETIC LEFT (SHL)

*Octal Code:* 7413

*Event Time:* 2

*Indicators:* OPR, FETCH, PAUSE

*Execution Time:* 3.0 microseconds + 0.5 microsecond for each shift

*Operation:* This instruction is used for scaling by shifting the combined content of the AC and MQ to the left one position more than the number of positions indicated by the content of core memory at address PC + 1 (or the next successive core memory location following the SHL command). During the shifting, zeros are shifted into vacated MQ11 positions. The L, AC, and MQ are treated as one long register during this operation. Bits shifted out of AC0 enter the L, and bits shifted out of the L are lost.

*Symbol:* Shift Y + 1 positions as follows:

ACj => ACj − 1
AC0 => L
MQ0 => AC11
MQj => MQj −1
0 => MQ11

## ARITHMETIC SHIFT RIGHT (ASR)

*Octal Code:* 7415

*Event Time:* 2

*Indicators:* OPR, FETCH, PAUSE

*Execution Time:* 3.0 microseconds + 0.5 microsecond for each shift.

*Operation:* This instruction is used for scaling and treats the AC and MQ as one long register. The combined content of the AC and the MQ is shifted right one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the ASR command). The sign bit, contained in AC0, enters vacated positions, the sign bit is preserved in the link, information shifted out of MQ11 is lost, and the L is set to correspond to the sign bit during this operation.

*Symbol:* Shift Y + 1 positions as follows:

AC0 = > L
AC0 = > AC0
ACj = > ACj + 1
AC11 = > MQ0
MQj = > MQj + 1

## LOGICAL SHIFT RIGHT (LSR)

*Octal Code:* 7417

*Event Time:* 2

*Indicators:* OPR, FETCH, PAUSE

*Execution Time:* 3.0 microseconds + 0.5 microsecond for each shift.

*Operation:* This instruction is used for scaling and treats the AC and MQ as one long register. The combined content of the AC and MQ is shifted right one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the LSR command). This command is similar to the ASR command except that zeros enter vacated positions instead of the sign bit entering these locations. Information shifted out of MQ11 is lost and the L is cleared during this operation.

*Symbol:* Shift Y + 1 positions as follows:

0 = > L
0 => AC0
ACj => ACj + 1
AC11 => MQ0
MQj => MQj + 1

## LOAD MULTIPLIER QUOTIENT (MQL)

*Octal Code:* 7421

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* This command clears the MQ, loads the content of the AC into the MQ, then clears the AC. This operation is essential to initializing any multiply or divide routine and can be combined with a MUY or DVI command to perform the operation just prior to executing a multiplication or a division using a 12-bit dividend.

*Symbol:* 0 => MQ
AC => MQ
0 => AC

## STEP COUNTER LOAD INTO ACCUMULATOR (SCA)

*Octal Code:* 7441

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the step counter is transferred into the AC. This command is used following an NMI command to establish the exponent of a normalized number to be used in floating point arithmetic. The AC should be cleared prior to issuing this command or the CLA command can be combined with the SCA to clear the AC then effect the transfer.

*Symbol:* SC V AC => AC

## MULTIPLIER QUOTIENT LOAD INTO ACCUMULATOR (MQA)

*Octal Code:* 7501

*Event Time:* 2

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The content of the MQ is transferred into the AC. This command is given to load the 12 least significant bits of the product into the AC following a multiplication or to load the quotient into the AC following a division. The AC should be cleared prior to issuing this command or the CLA command can be combined with the MQA to clear the AC then effect the transfer.

*Symbol:* MQ V AC => AC

## CLEAR ACCUMULATOR (CLA)

*Octal Code:* 7601

*Event Time:* 1

*Indicators:* OPR, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* The AC is cleared during event time 1, allowing this command to be combined with the other EAE commands that load the AC during event time 2 (such as SCA and MQA).

*Symbol:* 0 => AC

# Programming

## MULTIPLICATION

Multiplication is performed as follows:

1. Load the AC with the multiplier using the TAD instruction.
2. Transfer the content of the AC into the MQ using the MQL command.
3. Give the MUL command.

Note that steps 2 and 3 can be combined into one instruction.

The content of the MQ is then multiplied by the content of the next succesive core memory address (PC + 1). At the conclusion of the multiplication the most significant 12 bits of the product are held in the AC and the least significant 12 bits are held in the MQ. This operation takes a maximum of 21.0 microseconds, at the end of this time the next instruction is executed.

The following multiplication program examples indicate the operation of the Type 182 option in closed subroutines(routines which are incorporated into larger routines and are not written in a form which allows them to be called as a normal mathematical subroutine).

### Multiplication of 12-Bit Unsigned Numbers

Enter with a 12-bit multiplicand in AC and a 12-bit multiplier in core memory. Exit with high order half of product in a core memory location labeled HIGH, and with low order half of product in the AC. Program time is from 13.5 to 25.5 microseconds.

```
MQL  MUY      /LOAD MQ WITH MULTIPLICAND, INITIATE
              /MULTIPLICATION
MLTPLR        /MULTIPLIER
DCA  HIGH     /STORE HIGH ORDER PRODUCT
MQA           /LOAD AC WITH LOW ORDER PRODUCT
```

### Multiplication of 12-Bit Signed Numbers, 24-Bit Signed Product

Enter with a 12-bit multiplicand in AC and a 12-bit multiplier in core memory. Exit with signed 24-bit product in core memory locations designated HIGH and LOW. Program time is from 40.5 to 66.0 microseconds.

```
      CLL
      SPA                  /MULTIPLICAND POSITIVE?
      CMA CML IAC          /NO. FORM TWO'S COMPLEMENT
      MQL                  /LOAD MULTIPLICAND INTO MQ
      TAD     MLTPLR
      SPA                  /MULTIPLIER POSITIVE?
      CMA CML IAC          /NO. FORM TWO'S COMPLEMENT
      DCA     MLTPLR
      RAL
```

```
            DCA     SIGN        /SAVE LINK AS SIGN INDICATOR
            MUY                 /MULTIPLY
MLTPLR,     0                   /MULTIPLIER
            DCA     HIGH
            TAD     SIGN
            RAR                 /LOAD LINK WITH SIGN INDICATOR
            MQA
            SNL                 /IS PRODUCT NEGATIVE?
            JMP     LAST        /NO
            CLL CMA IAC         /YES
            DCA     LOW
            TAD     HIGH
            CMA
            SZL
            IAC
            DCA     HIGH
            SKP
LAST,       DCA     LOW
              .
              .
              .
```

## DIVISION

Division is performed as follows:

1. Load the 12 least significant bits of the dividend into the AC using the TAD instruction, then transfer the content of the AC into the MQ using the MQL command.
2. Load the 12 most significant bit of the dividend into the AC.
3. Give the DVI command.

The 24-bit dividend contained in the AC and MQ is then divided by the 12-bit divisor contained in the next successive core memory address (PC + 1). This operation takes a maximum of 36.5 microseconds and is concluded with a 12-bit quotient held in the MQ, the 12-bit remainder in the AC, and the link holding a 0 if divide overflow did not occur. To prevent divide overflow, the divisor in the core memory must be greater than the 12-bits of the dividend held in the AC. When divide overflow occurs, the link is set and the division is concluded after only one cycle. Therefore the instruction following the divisor in core memory should be an SZL microinstruction to test for overflow. The instruction following the SZL can be a jump to a subroutine that services the overflow. This subroutine can cause the program to type out an error indication, rescale the divisor or the dividend, or perform other mathematical corrections and repeat the divide routine.

The following division program examples indicate the operation of the Type 182 option in closed subroutines.

### Division of 12-Bit Unsigned Numbers

Enter with a 12-bit unsigned dividend in the AC and a 12-bit unsigned divisor in core memory. Exit with remainder in core memory location labeled REMAIN and with the quotient in the AC. Program time is a maximum of 44.0 microseconds.

```
CLL
MQL  DVI          /LOAD MQ, INITIATE DIVISION
DIVSOR            /DIVISOR
SZL               /OVERFLOW?
JMP               /YES, EXIT
DCA REMAIN
MQL               /LOAD AC WITH QUOTIENT
```

## Division of a 12-Bit Signed Numbers

Enter with a 12-bit signed dividend in the AC and a 12-bit signed divisor in core memory. Exit with unsigned remainder in core memory location REMAIN and a 12-bit signed quotient in the AC. Program time is a maximum of 65.0 microseconds.

```
CLL
SPA               /DIVIDEND POSITIVE?
CMA CML IAC       /NO
MQL
TAD .+11
SPA               /DIVISOR POSITIVE?
CMA CML IAC       /NO
DCA .+6
SNL               /QUOTIENT NEGATIVE?
CMA               /NO
CLL
DCA SIGN          /SET SIGN INDICATOR
DVI
DIVSOR            /DIVISOR
SZL               /OVERFLOW
JMP               /EXIT ON OVERFLOW
MQL
ISZ SIGN
CMA IAC
```

# CHAPTER 9

# AUTOMATIC RESTART TYPE KR01

This prewired option protects an operating program in the event of failure of the source of computer primary power. If a power failure occurs, this option causes a program interrupt and enables continued operation for 1 millisecond, allowing the interrupt routine to detect the power low condition as initiator of the interrupt, and to store the content of active registers (AC, L, MQ, etc.) and the program count in known core memory locations. When power is restored, the power low flag clears and a routine beginning in address 0000 starts automatically. This routine restores the content of the active registers and program counter to the conditions that existed when the interrupt occurred, then continues the interrupted program.

The KR01 option consists of three logic circuits:

A power interrupt circuit monitors the status signal of the computer power supply, and sets a power low flag when power is interrupted (due to a power failure or due to the operation of the POWER lock on the operator console). This flag causes a program interrupt when an interruption in computer power is detected.

A restart circuit assures that when a power interrupt occurs the logic circuits of the computer continue operation for 1 millisecond to allow a program subroutine to store the content of the active registers; maintains the inoperative condition of the computer during periods of power fluctuation; and clears the power low flag and restarts the program when power conditions are suitable for computer operation. A manual RESTART switch on the processor marginal-check frame enables or disables the automatic restart operation. With this switch in the ON (down) position, the option clears the program counter immediately and produces a signal to simulate operation of the START key on the operator console 200 milliseconds after power conditions are satisfactory. The PC is cleared so that operation restarts by executing the instruction in address 0000. This instruction is a JMP to the starting address of the subroutine which restores the content of the active registers and the program counter to the conditions that existed prior to the power low interrupt. The 200-millisecond delay assures that slow mechanical devices, such as Teletype equipment, have come to a complete stop before the program is resumed. Simulation of the manual START function causes the processor to generate a Power Clear pulse to clear internal controls and I/O device registers. With the RESTART switch in the OFF (up) position, the power low flag is cleared but the program must be started manually, possibly after resetting peripheral equipment or by starting the interrupted program from the beginning.

A skip circuit provides programmed sensing of the condition of the power low flag by adding the following instruction to the computer repertoire:

SKIP ON POWER LOW (SPL)

*Octal Code:* 6102

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the power low flag is sampled, and if it contains a 1 (indicating a power failure has been detected) the content of the PC is incremented by one so the next sequential instruction is skipped.

*Symbol:* If Power Low flag = 1, then PC +1 => PC

Since the time that operation of the computer can be extended after a power failure is limited to 1 millisecond, the condition of the power low flag should be the first status check made by the program interrupt subroutine. The beginning of the program interrupt subroutine, containing the SPL microinstruction and the power fail program sequence can be executed in 25.5 microseconds on a basic PDP-8 with an extended arithmetic element. The power fail program sequence stores the content of the active registers and program count in designated core memory locations, then relocates the calling instruction of the power restore subroutine to address 0000, as follows:

```
Address     Instruction          Remarks
0000        —                    /STORAGE FOR PC AFTER PROGRAM INTERRUPT
0001        JMP FLAGS            /INSTRUCTION EXECUTED AFTER PROGRAM
                                 /INTERRUPT
FLAGS,      SPL                  /SKIP IF POWER LOW FLAG = 1
            JMP OTHER            /INTERRUPT NOT CAUSED BY POWER LOW,
                                 /CHECK OTHER FLAGS
            DCA AC               /INTERRUPT WAS CAUSED BY POWER LOW,
                                 /SAVE AC
            RAR                  /GET LINK
            DCA LINK             /SAVE LINK
            MQA                  /GET MQ
            DCA MQ               /SAVE MQ
            TAD 0000             /GET PC
            DCA PC               /SAVE PC
            TAD RESTRT           /GET RESTART LOCATION
            DCA 0000             /DEPOSIT RESTART LOCATION IN 0000
            HLT
RESTRT      JMP ABCD             /ABCD IS LOCATION OF RESTART ROUTINE
```

Automatic program restart begins by executing the instruction stored in address 0000 by the power fail routine. The power restore subroutine restores the content of the active registers, enables the program interrupt facility, and continues the interrupted program from the point at which it was interrupted, as follows:

```
Address     Instruction          Remarks
0000        JMP ABCD
ABCD,       TAD MQ               /GET MQ
            MQL                  /RESTORE MQ
            TAD LINK             /GET LINK
            CLL RAL              /RESTORE LINK
            TAD AC               /RESTORE AC
            ION                  /TURN ON INTERRUPT
            JMP I PC             /RETURN TO INTERRUPTED PROGRAM
```

# SECTION B

# INPUT-OUTPUT EQUIPMENT

# CHAPTER 1

# BASIC IOT PROGRAMMING

Three basic modes of programming can be used to transfer information or control signals between input/output devices and the PDP-8. These modes are programmed data transfers, program interrupt, and data break transfers. To understand the use which can be made of each of these control modes assume that the PDP-8 is connected as the primary control element in a system which contains several furnaces or kilns, continuous-belt conveyors which transport products through the furnaces, a visual monitor panel, and a drum memory. Assume that each furnace or zone of an oven contains a controller which has a digital temperature readout, and overtemperature and undertemperature alarms which can be set by a digital readin. Next, assume that the conveyor for each oven is automatically loaded with blocks of products spaced at given distances on the conveyor and that the conveyor has a normal speed, a high emergency speed, and a stop control. Finally, assume that the visual monitor and drum memory periodically cause data breaks to transfer temperature and controls status information from core memory into visual display devices and drum storage. The following explanation of the use of each of the three control modes in programming this hypothetical system can easily be translated into examples of data processing or other control system programs.

## Programmed Data Transfers

All peripheral equipment transfers information to or from the computer by programmed instructions. This means of communication can be used as the sole method of transferring information or can be used to initialize equipment using the program interrupt or data break facilities. This mode of operation utilizes the IOT instruction which is divided into three parts. Bits 0, 1, and 2 contain an operation code of 6 to specify the IOT microinstruction. Bits 3 through 8 serve as a device selection code which is transmitted to all peripheral equipment and which activates only the equipment designated by a specific code number contained within these bits. Bits 9, 10, and 11 control the IOP generator within the processor and enable or disable generation of IOP1, IOP2, and IOP4 pulses during each IOT instruction. A device selector within each peripheral equipment monitors the device selection lines and enables pulse amplifiers when its assigned select code has been detected within bits 3 through 8 of an IOT instruction. When enabled in this manner the pulse amplifiers produce positive or negative IOT pulses when triggered by an associated IOP pulse. The IOT pulses, in turn, perform data transfers to or from the computer or perform control functions within the peripheral equipment.

Each peripheral equipment can contain one or more device selector. A device selector can consist of a Type 4605 Pulse Amplifier system module, a Type W103 Device Selector FLIP CHIP module, or can be constructed of three FLIP-CHIP modules such as the Type R603 Pulse Amplifier, R111 Diode Gate, and R002 Diode Cluster. Regardless of its circuit components, a device selector consists of a 6-input negative diode NAND gate which is enabled only when the select code of the specified device is contained in the instruction. The output from this NAND gate enables gating circuits at the input of each of three pulse amplifiers which are triggered by the IOP1, IOP2, or IOP4 pulse.

## SENSE FOR DEVICE READY

In preparation for a normal data transfer the computer program normally checks the ready status of the transmitting or receiving device by means of a skip instruction. This skip instruction can skip on either the ready or not ready status of the device, depending upon the internal operations of that device. In our imaginary PDP-8 control system, assume that the control for each furnace contains an "up to temperature flag" and an associated instruction which provides a Skip pulse to the program counter control element when the operating temperature range has been attained. Therefore the programming can contain this skip on-up-to-temperature instruction followed by a JMP instruction which transfers program control back in a loop so that it repeatedly checks the up to temperature flag until the acceptable temperature range has been attained. Then the next step in the program proceeds with an operation, such as initiate loading of the conveyor and movement of the conveyor at the normal speed. In some instances the program will proceed to assemble data so that it can be transferred into or out of the computer.

## ASSEMBLE DATA

The ready control described under the previous heading can indicate that a device is operating in a known control mode or that data is ready for transmission. In our example the former condition is true so that data assembly follows the sense for ready operation. To assemble data the program issues commands in one or more IOT instructions to clear and load a buffer with data to be transferred from the accumulator, or to clear and load the accumulator for data to be transferred to a peripheral equipment buffer. In the imaginary control system described earlier, data assembly could consist of an instruction which generates an IOT1 pulse to clear an information buffer, generates an IOT2 pulse to load a temperature setting into the six least significant bits of this buffer and loads the status of the undertemperature and overtemperature flags into the two most significant bits of this buffer.

## EFFECT A TRANSFER

Actual data transfer between peripheral equipment and the PDP-8 accumulator is performed by an IOT instruction. Usually a transfer into the PDP-8 is performed by an instruction in which an IOT1 pulse is generated to clear the accumulator and a later IOT pulse is generated to strobe the content of an external buffer into the accumulator. In transfers from the accumulator to an external buffer the static accumulator data lines are used to condition gates at the input of a static buffer, then an IOT pulse is generated to actuate the gates and transfer the static conditions into the buffer. In the case of our hypothetical process control system an IOT instruction can be developed to clear the accumulator, then read the content of the data buffer into the accumulator.

The entire sequence of operations described under Programmed Data Transfers is summarized in the following program example, using legitimate memory reference and operate instructions, and using artificial IOT instructions.

```
                    CLA
                    TAD         /LOAD OVERTEMPERATURE VALUE INTO AC
                    LOT         /CLEAR AND LOAD OVERTEMPERATURE SET
Initialize                      /POINT
                    CLA
                    TAD         /LOAD UNDERTEMPERATURE VALUE INTO AC
                    LUT         /CLEAR AND LOAD UNDERTEMPERATURE SET
                                /POINT
                    TON         /TURN ON FURNACE
Sense for           SUT         /SKIP ON DEVICE #1 UP TO TEMP.
device ready        JMP -1      /LOOP IF NOT UP TO TEMP
(one or more        IPC         /INITIATE PRODUCT TRANSPORT TO CONVEYOR
devices)                        /AND CONVEYOR OPERATION AT NORMAL
                                /SPEED
Assemble
Data                LDB         /CLEAR AND LOAD DEVICE BUFFER WITH DATA
                                /AND STATUS
Effect Transfer     RDB         /CLEAR AC AND READ DATA BUFFER INTO AC
                    RAL         /ROTATE OVERTEMP. CONTROL STATUS FROM
                                /AC0 INTO L
                    SNL OR SZL  /SENSE OVERTEMP. CONTROL
                    JMS         /JUMP TO OVERTEMP. SUBROUTINE WHICH
                                /TURNS OFF FURNACE, STOPS PRODUCT
                                /LOADING INTO CONVEYOR, ADVANCES
                                /CONVEYOR AT EMERGENCY SPEED TO
                                /REMOVE PRODUCTS FROM FURNACE, THEN
                                /STOPS CONVEYOR
Process             RAL         /ROTATE UNDERTEMP. CONTROL STATUS
Data                            /INTO L
                    SNL OR SZL  /SENSE UNDERTEMP. CONTROL
                    JMS         /JUMP TO SUBROUTINE WHICH STOPS
                                /CONVEYOR MOTION AND JUMPS BACK TO
                                /THE BEGINNING OF THE MAIN ROUTINE TO
                                /WAIT FOR UP TO TEMP.
                    RTR         /RELOCATE DATA
                    DCA         /STORE TEMPERATURE DATA
```

# Program Interrupt

Urgent requirements for programmed data transfer or programmed control functions by peripheral equipment can be satisfied through use of the program interrupt facility. This facility allows an external device to cause the main computer program to be interrupted and a subroutine to be initiated to service the interrupting device. Use of this facility simplifies basic programming by eliminating the need for checking alarm conditions and allows the alarm conditions themselves to activate corrective operations, rather than waiting for cyclic checking by the main routine.

When the program interrupt feature is used address 0001 is automatically specified as the first address of a subroutine that checks and services the interrupt condition. Usually the instruction stored in this address is a jump to a location where the subroutine really begins. As designated in Chapter 4 of Section A of this handbook the

program interrupt subroutine must locate the device causing the interruption, take some corrective action, restore or enable the program interrupt synchronization element of the computer by execution of an ION instruction, and return program control to the main program at the point at which the interrupt occurred. If only one device is connected to the program interrupt facility no checking is required to locate the interrupting device. However, if many devices are connected to the program interrupt bus (as normally is the case) the interrupt subroutine must perform repeated skip instructions to test the condition of the various devices. This testing should be accomplished by a program-established priority system so that the devices which need servicing in the least amount of time or which require servicing most frequently are checked first, depending upon the application.

In our hypothetical process control system if the overtemperature alarms for each furnace are connected to the program interrupt bus the interrupt subroutine can skip on the condition of the overtemperature flag to a portion of the routine which de-energizes the appropriate furnace and performs any required operations in the control of the conveyor for that oven (such as inhibiting additional loading of the conveyor, removing products from the oven by high speed advance of the conveyor, or by initiating other shut down procedures). The sequence of testing for the furnace causing the overtemperature alarm can proceed from the first furnace to the last furnace if it is most important to prevent unfired products from entering the defective oven, can be performed from the last to the first furnace if over firing cannot occur, or can be performed in the sequence determined by the various furnace temperatures.

# Data Break Transfers

Peripheral equipment requiring rapid or periodic data transfers to or from the core memory of the PDP-8 can be connected to use the data break facilities. Where more than one such input/output device is used in the computer system they must be connected to the data break facility through a multiplexer switch (such as the Type DM01 option) which assigns a pre-established priority to each device. This facility allows the main computer program to be suspended for a time while individual or block data transfers occur between the memory buffer register and the peripheral equipment. In some cases these transfers occur to or from blocks of sequential core memory addresses or occur individually, but each transfer occurs at a device-specified address. Where individual transfers occur sporadically in time, each transfer is interleaved with portions of the main program. Following each data break the transferring equipment can issue a program interrupt to enter a sub-routine which reinitializes the device with the transfer direction and address information required for the succeeding break.

In our hypothetical process control system it can be assumed that a real-time clock in the visual display produces a data break periodically to transfer temperature and control status information from specific core memory addresses into visual readout or display devices. The drum memory can be assumed to initiate data breaks to read and store data from core memory so that it can be analyzed for quality control evaluation after running the heat. The frequency of the data break requests performed in this manner must be related to the program timing required to read this information into core memory from the temperature controlling devices, assuming the maximum amount of time for any program interrupts and data breaks during the program.

The data break is accomplished by supplying Break Request and Cycle Select signals to the major state generator, supplying a Transfer Direction signal to the memory buffer register control element, supplying an address to the memory address register,

and (for a transfer into the PDP-8) supplying a data word to the memory buffer register. When a single-cycle data break is requested in this manner, at the conclusion of the current instruction the Break state is entered to transfer a data word. When the direction of transfer is into the PDP-8, the data word must be supplied the memory buffer register within the first half of the Break cycle. When the direction of transfer is out of the PDP-8, the data word is available for strobing by the device approximately 350 microseconds after entry into the Break state. When a three-cycle data break is requested, at the conclusion of the current instruction the Word Count state is entered to commence the data break; but the transfer does not occur until the the third (Break) cycle of the data break.

Timing and data requirements of the input/output device determine the number of consecutive data breaks that occur before control of computer operations is returned to the program. One Break cycle is required for each data word transfer, so very fast devices which require blocks of information can maintain the Break Request signal to perform consecutive data breaks until a device-specified block length transfer is completed. Devices using consecutive data breaks must synchronize their operations to the speed of the computer to transfer words with core memory at the single-cycle rate of 666 kc (one word every 1.5 microseconds) or at the three-cycle rate of 222 kc (one word every 4.5 microseconds). Normal clock pulses used with the computer are available to synchronize the operation of peripheral equipment with the PDP-8 during a data break. Slower equipment using the data break facilities must initiate a separate data break for each word transfer.

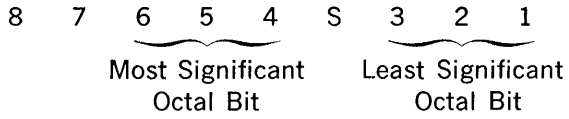
# CHAPTER 2

# TELETYPE AND CONTROL

## Teletype Model 33 ASR

The standard Teletype Model 33 ASR (automatic send-receive) can be used to type in or print out information at a rate of up to ten characters per second, or to read in or punch out perforated paper tape at a ten characters per second rate. Signals transferred between the 33 ASR and the control logic are standard serial, 11 unit code Teletype signals. The signals consist of marks and spaces which correspond to idle and bias current in the Teletype,and to zeros and ones in the control and computer. The start mark and subsequent eight character bits are one unit of time duration and are followed by the stop mark which is two units.

The 8-bit code used by the Model 33 ASR Teletype unit is the American Standard Code for Information Interchange (ASCII) modified. To convert the ASCII code to Teletype code add 200 octal (ASCII + $200_8$ = Teletype). This code is read in the reverse of the normal octal form used in the PDP-8 since bits are numbered from right to left, from 1 through 8, with bit 1 having the least significance. Therefore perforated tape is read:




The Model 33 ASR set can generate all assigned codes except 340 through 374 and 376. Generally codes 207, 212, 215, 240 through 337, and 377 are sufficient for Teletype operation. The Model 33 ASR set can detect all characters, but does not interpret all of the codes that it can generate as commands. The standard number of characters printed per line is 72. The sequence for proceeding to the next line is a carriage return followed by a line feed (as opposed to a line feed followed by a carriage return). Appendix 2 lists the character code for the Teletype. Punched tape format is as follows:

| | Tape Channel | | | |
|---|---|---|---|---|
| | 87 | 654 | S | 321 |
| Binary Code (Punch = 1) | 10 | 110 | | 100 |
| Octal Code | 2 | 6 | | 4 |

## Teletype Control

Serial information read or written by the Teletype unit is assembled or disassembled by the control for parallel transfer to the accumulator of the processor. The control also provides the program flags which cause a program interrupt or an instruction skip based upon the availability of the Teletype and the processor as a function of the program.

In all programmed operation, the Teletype unit and control are considered as a Teletype in (TTI) as a source of input intelligence from the keyboard or the perforated-tape reader and is considered a Teletype out (TTO) for computer output information to be printed and/or punched on tape. Therefore, two device selectors are used; the select

code of 03 initiates operations associated with the keyboard/reader, and the device selector, assigned the select code of 04, performs operations associated with the teleprinter/punch. Parallel input and output functions are performed by corresponding IOT pulses produced by the two device selectors. Pulses produced by IOP1 pulse trigger skip gates; pulses produced by the IOP2 pulse clear the control flags and/or the accumulator; and pulses produced by the IOP4 pulse initiate data transfers to or from the control.

# Keyboard/Reader

The keyboard and tape reader control contains an 8-bit buffer (TTI) which assembles and holds the code for the last character struck on the keyboard or read from the tape. Teletype characters from the keyboard/reader are received serially by the 8-bit shift register TTI. The character code of a Teletype character is loaded into the TTI so that spaces correspond with binary zeros and marks correspond to binary ones. Upon program command the content of the TTI is transferred in parallel to the accumulator. When a Teletype character starts to enter the TTI the control de-energizes a relay in the Teletype unit to release the tape feed latch. When released, the latch mechanism stops tape motion only when a complete character has been sensed, and before sensing of the next character is started. A keyboard flag is set to one, and causes a program interrupt when an 8-bit computer character has been assembled in the TTI from a Teletype character. The program senses the condition of this flag with a KSF microinstruction and issues a KRB microinstruction which clears the AC, clears the keyboard flag, transfers the content of the TTI into the AC, and enables advance of the tape feed mechanism. Instructions for use in supplying data to the computer from the Teletype are:

SKIP ON KEYBOARD FLAG (KSF)

*Octal Code:* 6031

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The keyboard flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Keyboard Flag = 1, then PC +1 => PC

CLEAR KEYBOARD FLAG (KCC)

*Octal Code:* 6032

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Both the AC and the keyboard flag are cleared in preparation for transferring a Teletype character into the AC.

*Symbol:* 0 => AC
0 => Keyboard Flag

READ KEYBOARD BUFFER STATIC (KRS)

*Octal Code:* 6034

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the TTI is transferred into bits 4 through 11 of the AC. This is a static command in that neither the AC nor the keyboard flag is cleared.

*Symbol:* TTI V AC 4-11 => AC 4-11

READ KEYBOARD BUFFER DYNAMIC (KRB)

*Octal Code:* 6036

*Event Time:* 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The AC and the keyboard flag are both cleared, then the content of the TTI is transferred into bits 4 through 11 of the AC.

*Symbol:* 0 => AC, Keyboard Flag
TTI V AC 4-11 => AC 4-11

A program sequence loop to read input information into the computer from the Teletype keyboard or tape reader can be written as follows:

```
LOOK,    KSF          /SKIP WHEN TTI IS FULL
         JMP  LOOK
         KRB          /READ TTI INTO AC
```

# Teleprinter/Punch

Eight-bit computer characters from the accumulator are loaded in parallel into the 8-bit flip-flop shift register TTO for transmission to the Teletype unit. The control generates the start space, then shifts the eight character bits into the printer selector magnets of the Teletype unit, and then produces the stop mark. This transfer of information from the TTO into the Teletype unit is accomplished in a serial manner at the normal Teletype rate. A teleprinter flag in the teleprinter control is set when the last bit of the Teletype code has been sent to the teleprinter/punch, indicating that the TTO is ready to receive a new character from the AC. The flag is connected to both the program interrupt synchronization element and the PC control (instruction skip) element. Upon detecting the set (binary one) condition of the flag by means of the TSF microinstruction the program issues a TLS microinstruction which clears the flag and loads a new computer character into the TTO.

The instruction list for printing or punching is:

SKIP ON TELEPRINTER FLAG (TSF)

*Octal Code:* 6041

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The teleprinter flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Teleprinter Flag = 1, then PC +1 => PC

*Octal Code:* 6042 CLEAR TELEPRINTER FLAG (TCF)

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The teleprinter flag is cleared to 0.

*Symbol:* 0 => Teleprinter Flag

*Octal Code:* 6044 LOAD TELEPRINTER AND PRINT (TPC)

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The TTO is loaded from the content of bits 4 through 11 of the AC; then the Teletype character just loaded is selected, and punched and/or printed.

*Symbol:* AC 4-11 => TTO

*Octal Code:* 6046 LOAD TELEPRINTER SEQUENCE (TLS)

*Event Time:* 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The teleprinter flag is cleared; then a Teletype character code is transferred from the content of AC 4-11 into the TTO, the character is selected and punched and/or printed.

*Symbol:* 0 => Teleprinter Flag
AC 4-11 => TTO

A program sequence loop to print and/or punch a character when the TTO is free can be written as follows:

```
FREE,    TSF          /SKIP WHEN FREE
         JMP FREE
         TLS          /LOAD TTO,  PRINT OR PUNCH
```

# Teletype System Type LT08

The Teletype facility of the basic computer can be expanded to accommodate several Model 33 or Model 35 Automatic Send Receive or Keyboard Send Receive units by addition of the Type LT08 option. Each Teletype line added to the PDP-8 system contains logic elements that are functionally identical to those of the basic Teletype control. Therefore, instructions and programming for each line of an LT08 equipment are similar to those described previously for the basic Teletype unit, The following device select codes have been assigned for five lines of LT08 equipment:

| Line Unit | Select Codes |
|---|---|
| 1 | 40 and 41 |
| 2 | 42 and 43 |
| 3 | 44 and 45 |
| 4 | 46 and 47 |
| 5 | 11 and 12 |

Instruction mnemonics for Teletype equipment in the LT08 system are not recognized by the program assembler (PAL III) and must be defined by the programmer. Mnemonic codes can be defined by the mnemonic code of the comparable basic Teletype microinstruction, suffixed with "LT" and the line number. For example, the following instructions can be defined for line 3:

| Mnemonic | Octal | Operation |
|---|---|---|
| TSFLT3 | 6441 | Skip if teleprinter 3 flag is a 1. |
| TCPLT3 | 6442 | Clear teleprinter 3 flag. |
| TPCLT3 | 6444 | Load teleprinter 3 buffer (TTO3) from the content of AC4-11 and print and/or punch the character. |
| TLSLT3 | 6446 | Load TTO3 from the content of AC4-11, clear teleprinter 3 flag, and print and/or punch the character. |
| KSFLT3 | 6451 | Skip if keyboard 3 flag is a 1. |
| KCCLT3 | 6452 | Clear AC and clear keyboard 3 flag. |
| KRSLT3 | 6454 | Read keyboard 3 buffer (TTI3) static. The content of TTI3 is loaded into AC4-11 by an OR transfer. |
| KRBLT3 | 6456 | Clear the AC, clear keyboard 3 flag, and read the content of TTI3 into AC4-11. |

# *UNIVERSITY INTRAMURAL CORRESPONDENCE*

TO: Users of the PDP-8

FROM: Michael S. Wolfberg

DATE: September 1, 1966 (corrected)

SUBJECT: LT08 Teletype System

The PDP-8 is equipped with an LT08 Teletype System for the control of remote Teletypes. The present configuration includes two line units, but only Line Unit 1 is currently connected to a Dataphone. Its corresponding instructions are listed below.

| Mnemonic | Octal | | |
|---|---|---|---|
| TSFLT1 | 6411 | TSFLT2 | 6431 |
| TCPLT1 | 6412 | TCPLT2 | 6432 |
| TPCLT1 | 6414 | TPCLT2 | 6434 |
| TLSLT1 | 6416 | TLSLT2 | 6436 |
| KSFLT1 | 6401 | KSFLT2 | 6421 |
| KCCLT1 | 6402 | KCCLT2 | 6422 |
| KRSLT1 | 6404 | KRSLT2 | 6424 |
| KRBLT1 | 6406 | KRBLT2 | 6426 |

NOTE: THE ABOVE LIST IS CORRECT. THE LIST GIVEN IN THE PDP-8 USERS HANDBOOK (5/66) IS INCORRECT.

Each line unit of the LT08 Teletype System is functionally identical to the on-line Teletype control. Therefore, instructions and programming for each line are similar to those for the on-line Teletype.

# CHAPTER 3

# HIGH SPEED PERFORATED TAPE READER AND CONTROL TYPE 750C

This device senses 8-hole perforated paper or Mylar tape photoelectrically at 300 characters per second. The reader control requests reader movement, transfers data from the reader into the reader buffer (RB), and signals the computer when incoming data is present. Reader tape movement is started by a reader control request to simultaneously release the brake and engage the clutch. The 8-bit reader buffer sets the reader flag to 1 when it has been filled from the reader and transfers data into bits 4 through 11 of the accumulator under program control. The reader flag is connected to the computer program interrupt and instruction skip facilities, and is cleared by IOT pulses. Tape format is as described for the Teletype unit. Computer instructions for the reader are:

SKIP ON READER FLAG (RSF)

*Octal Code:* 6011

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The reader flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Reader Flag = 1, then PC +1 => PC

READ READER BUFFER (RRB)

*Octal Code:* 6012

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the reader buffer is transferred into bits 4 through 11 of the AC and the reader flag is cleared. This command does not clear the AC.

*Symbol:* RB V AC 4-11 => AC 4-11
0 => Reader Flag

READER FETCH CHARACTER (RFC)

*Octal Code:* 6014

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The reader flag and the reader buffer are both cleared, one character is loaded into the reader buffer from tape, and the reader flag is set when this operation is completed.

*Symbol:* 0 => Reader Flag, RB
Tape Data => RB
1 => Reader Flag when done

A program sequence loop to read a character from perforated tape can be written as follows:

```
        RFC          /FETCH CHARACTER FROM TAPE
LOOK,   RSF          /SKIP WHEN RB FULL
        JMP LOOK
        CLA
        RRB          /LOAD AC FROM RB
```

# CHAPTER 4

# HIGH SPEED TAPE PUNCH CONTROL TYPE 75E

This option consists of a Teletype BRPE paper tape punch that perforates 8-hole tape at a rate of 63.3 characters per second. Information to be punched on a line of tape is loaded in an 8-bit punch buffer (PB) from AC bits 4 through 11. The punch flag becomes a 1 at the completion of punching action, signaling that new information may be transferred into the punch buffer, and punching initiated. The punch flag is connected to the computer program interrupt and instruction skip facility. Tape format is as described in Chapter 2. The punch instructions are:

### SKIP ON PUNCH FLAG (PSF)

*Octal Code:* 6021

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The punch flag is sensed, and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Punch Flag = 1, then PC + 1 => PC

### CLEAR PUNCH FLAG (PCF)

*Octal Code:* 6022

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Both the punch flag and the punch buffer are cleared in preparation for receiving a new character from the computer.

*Symbol:* 0 => Punch Flag, PB

### LOAD PUNCH BUFFER AND PUNCH CHARACTER (PPC)

*Octal Code:* 6024

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* An 8-bit character is transferred from bits 4 through 11 of the AC into the punch buffer and then this character is punched. This command does not clear the punch flag or the punch buffer.

*Symbol:* AC4-11 V PB => PB

## LOAD PUNCH BUFFER SEQUENCE (PLS)

*Octal Code:* 6026

*Event Time:* 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The punch flag and punch buffer are both cleared, the content of bits 4 through 11 of the AC is transferred into the punch buffer, the character in the PB is punched in tape, and the punch flag is set when the operation is completed.

*Symbol:* 0 => Punch Flag, PB
AC4-11 => PB
1 => Punch Flag when done

A program sequence loop to punch a character when the punch buffer is "free" can be written as follows:

```
FREE,    PSF          /SKIP WHEN FREE
         JMP FREE
         PLS          /LOAD PB FROM AC AND PUNCH
                      /CHARACTER
```

# CHAPTER 5

# ANALOG-TO-DIGITAL CONVERTER

# TYPE 189

This converter operates in the conventional successive approximation manner, using the memory buffer register as a distributor shift register and using the accumulator as the digital buffer register. Converter operation is initiated by an IOT command that produces a Pause pulse (as most IOT commands do) and starts the conversion process. With the AC cleared, this process starts by asuming that the value of the analog input signal is at mid scale by setting a binary 1 into the most significant bit of the accumulator and producing a voltage equal to the center of the input range of the converter (ground to −10 volts). This voltage is produced by a digital-to-analog converter which operates as a function of a number contained in the accumulator. This voltage is then compared with the analog input signal and the result of the comparison is used to clear the most significant bit of the accumulator if the approximated voltage generated is of greater amplitude than the analog input signal. This process is then repeated by setting the next least significant bit of the accumulator to the 1 state, generating the analog signal according to the content of the accumulator, and comparing this signal with the analog input signal to clear the bit of the AC which was just set previously if the generated signal is greater than the input signal being measured. This process is repeated a number of times depending upon the prewired accuracy of the conversion. Each approximation reduces the error of the resultant binary number in the AC by approximately one half. The bit of the accumulator which is first set and then evaluated, is controlled by the memory buffer register. During the first approximation, a binary 1 is set into most significant bit of the MB and is shifted right one place at the conclusion of each approximation. The bit of the accumulator which is processed is determined by the location of this binary 1 in the MB. Sensing of the location of this binary 1 in the MB is also used to control the number of approximations performed, and hence determines the accuracy of the conversion. Since the conversion is started at the time the binary 1 is shifted in the MB, one conversion takes place after the sensing of the 1 in the MB which discontinues the conversion process. At the conclusion of the conversion a Restart pulse is produced by the converter which clears the MB and continues the normal computer program. At this time the digital equivalent of the analog input signal is contained in the accumulator as a 12-bit unsigned binary number. Insignificant magnitude bits can be rotated out of the AC by an instruction such as 7110 (RAR and CLL).

To save program running time, the converter should be adjusted to provide only the accuracy required by the program application. Maximum error of the converter is equal to the switching point error plus the quantization error. Maximum quantization error is equal to the binary value of the least significant bit. Switching point error and total conversion time are functions of the adjusted accuracy of the converter as indicated in Table 1.

TABLE 1 ANALOG-TO-DIGITAL CONVERTER TYPE 189 CHARACTERISTICS

| Adjusted Bit Accuracy | Switching Point Error (in per cent) | Conversion Time per Bit (in microseconds) | Total Conversion Time (in microseconds) | Instruction Execution Time (in microseconds) |
|---|---|---|---|---|
| 6 | ±1.6 | 1.0 | 6 | 7.6 |
| 7 | ±0.8 | 1.85 | 13 | 14.6 |
| 8 | ±0.4 | 2.5 | 20 | 21.6 |
| 9 | ±0.2 | 2.7 | 24 | 25.6 |
| 10 | ±0.1 | 2.7 | 27 | 28.6 |
| 11 | ±0.05 | 4.1 | 45 | 46.6 |
| 12 | ±0.025 | 4.6 | 55 | 56.6 |

The ADC is the only instruction associated with the Type 189 converter.

## CONVERT ANALOG TO DIGITAL (ADC)

*Octal Code:* 6004

*Event Time:* Not applicable

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* This time is related to the adjusted converter accuracy as listed in Table 1.

*Operation:* The analog input signal is converted to an unsigned digital value which is held in the AC at the end of the conversion.

*Symbol:* None

# CHAPTER 6

# ANALOG-TO-DIGITAL CONVERTER TYPE 138E AND MULTIPLEXER CONTROL TYPE 139E

The Type 138E/139E General-Purpose Analog-to-Digital Converter and Multiplexer Control combines a versatile, multipurpose converter with a multiplexer to provide a fast, automatic, multichannel scanning and conversion capability. It is intended for use in systems in which computers sample and process analog data from sensors or other external signal sources at high rates. For example, analog data on each of 64 channels can be accepted and converted into 12-bit digital numbers 415 times per second.* Switching point accuracy in this instance is 99.975 per cent, with an additional quantization error of half the least significant bit (LSB). If less resolution and accuracy is required, all 64 channels can be scanned and the analog signals on them converted into 6-bit digital numbers 1,360 times each second.** Switching point accuracy in this case is 99.2 per cent, again with the additional quantization error of half the digital value of the LSB.

The Type 139 Multiplexer Control can include from 1 to 32 series A100 Multiplexer modules determined by the user. Each module addresses one of two channels for a maximum of 64 channels per Type 139E control. In the Individual Address mode, the Type 139 routes the data from any selected channel to the Type 138E converter input. In the Sequential Address mode, the multiplexer advances its channel address by one each time it receives an increment command, returning to channel zero after scanning the last channel. Sequenced operations can be short-cycled when the number of channels in use is less than the maximum available.

*Conversion rate $=[(35+2.5)\ (10^{-6})\ (64)]^{-1}=415$ cycles/sec
**Conversion rate $=[(9+2.5)\ (10^{-6})\ (64)]^{-1}=1360$ cycles/sec

TABLE 2 ANALOG-TO-DIGITAL CONVERTER TYPE 139E CHARACTERISTICS

| Word Length (in bits) | Switching Point Error*** (in percent) | Total Conversion Time (in microseconds) | Conversion Rate (in kc) |
|---|---|---|---|
| 6 | ±1.6 | 9.0 | 110.0 |
| 7 | ±0.8 | 10.5 | 95.0 |
| 8 | ±0.4 | 12.0 | 83.0 |
| 9 | ±0.2 | 13.5 | 74.0 |
| 10 | ±0.1 | 17.0 | 58.5 |
| 11 | ±0.05 | 25.0 | 40.0 |
| 12 | ±0.025 | 35.0 | 28.5 |

***±½ LSB for quantizing error.

The Type 138E is a successive approximation converter that measures a 0 to 10 volt analog input signal and provides a binary output indication of the amplitude of the input signal. Output indication accuracy is a function of the conversion time, and is determined by a switch on the front panel. Each of the seven positions of the rotary switch establishes an output word length, conversion accuracy, and conversion time for operation of the converter. Overall conversion error equals switching point error plus a quantization of ±½ the digital value of the LSB. Converter characteristics selected for each switch position are specified in Table 2.

## CONVERTER SPECIFICATIONS

*Monotonicity:* Guaranteed for all settings

*Aperture Time:* Same as conversion time

*Converter Recovery Time:* None

*Analog Input:* 0 to −10 volts is standard. Bipolar or specific amplitude range input can be accommodated on special request. If a different voltage range is desired, it is recommended that an amplifier be used at the source, since this will also provide a low driving impedance and reduce the possibilities of noise pickup between the source and the converter.

*Input Loading:* ±1 microampere and 125 picofarads for the standard 0 to −10 volt input.

*Digital Output:* A signed 6- to 12-bit binary number in 2's complement notation. A 0 volt input yields a digital output number of $4000_8$; a −5 volt input produces $0000_8$; and a −10 volt input gives an output of $3777_8$.

*Controls:* Binary readout indicators and a seven-position rotary switch for selecting converter characteristics are provided on the front panel.

The Type 139E Multiplexer Control is intended for use with the Type 138E or Type 189 analog-to-digital conversion systems in applications where the PDP-8 must process sampled analog data from multiple sources at high speeds. Under program control the multiplexer can select from 2 to 64 analog input signal channels for connection to the input of an analog-to-digital converter. Channel selection is provided by Type A100, A101, A102, or A103 Multiplex Switch FLIP CHIP modules. These module types each have slightly different timing, impedance, and power characteristics so that multiplexers can be built for wide differences in application by selecting the appropriate module type. Each module contains two independent, floating, transistor switches letting the user select any multiple of two channels to a maximum of 64. In the individual address mode, the Type 139E routes the analog data from any program-selected channel to the converter input. In the sequential address mode, the multiplexer advances the channel address by one each time it receives an incrementing command, returning to channel zero after scanning the last channel. Sequenced operations can be short-cycled when the number of channels in use is less than the maximum available.

A 6-bit channel address register (CAR) specifies a channel number from $0\text{-}77_8$. A channel address may be chosen in one of two ways. It can be specified by the content of bits 6-11 of the AC or by incrementing the content of the CAR.

## MULTIPLEXER SPECIFICATIONS

*Indicators:* Six binary indicators on the front panel give visual indication of the selected channel.

*Multiplexer Switching Time:* The time required to switch from one channel to any program-specified channel, or to select the next adjacent channel when the content of the CAR is incremented is 2.5 microseconds. This time is measured from when either a select or increment command is received.

Both the converter and multiplexer circuits are constructed entirely of FLIP CHIP modules. Both the Type 138E converter and the Type 139E Multiplex Control (implemented to 24 input channels) circuits can be contained in one standard 64-connector module mounting panel.

The following IOT commands have been assigned to the Type 138E/139E converter system:

SKIP ON A-D FLAG (ADSF)

*Octal Code:* 6531

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The A-D converter flag is sensed, and if it contains a binary 1 (indicating that the conversion is complete) the content of the PC is incremented by one so that the next instruction is skipped.

*Symbol:* If A-D Flag = 1, then PC + 1 => PC

CONVERT ANALOG VOLTAGE TO DIGITAL VALUE (ADCV)

*Octal Code:* 6532

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* This time is a function of the accuracy and word length switch setting as listed in Table 2.

*Operation:* The A-D converter flag is cleared, the analog input voltage is converted to a digital value, and then the A-D converter flag is set to 1. The number of binary bits in the digital-value word and the accuracy of the word is determined by the preset switch position.

*Symbol:* 0 => A-D Flag at start of conversion, then
1 => A-D Flag when conversion is done.

READ A-D CONVERTER BUFFER (ADRB)

*Octal Code:* 6534

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The converted number contained in the converter buffer (ADCB) is transferred into the AC as a normalized word (shifted into the most significant bits), unused bits of the AC are cleared, and the A-D converter flag is cleared.

*Symbol:* ADCB => AC
0 => A-D Converter Flag

CLEAR MULTIPLEXER CHANNEL (ADCC)

*Octal Code:* 6541

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The channel address register (CAR) of the multiplexer is cleared in preparation for setting of a new channel.

*Symbol:* 0 => CAR

## SET MULTIPLEXER CHANNEL (ADSC)

*Octal Code:* 6542

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The channel address register of the multiplexer is set to the channel specified by bits 6 through 11 of the AC. A maximum of 64 single-ended or 32 differential input channels can be used.

*Symbol:* AC 6-11 => CAR

## INCREMENT MULTIPLEXER CHANNEL (ADIC)

*Octal Code:* 6544

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the channel address register of the multiplexer is incremented by one. If the maximum address is contained in the register when this command is given, the minimum address (00) is selected.

*Symbol:* CAR +1 => CAR

A program to cycle through all channels of the converter a given number of times, storing the conversion values at successive core memory locations can be written as follows:

```
LOOP,   ADIC          /INCREMENT CAR
        ADCV          /INITIATE CONVERSION
        ADSF          /WAIT FOR FLAG
        JMP.-1
        ADRB          /READ A-D CONVERTER BUFFER
        DCA I Z 10    /STORE RESULT IN ADDRESS SPECIFIED
                      /BY AUTO-INDEX REGISTER 10
        ISZ CNTR      /INCREMENT CYCLE COUNTER
        JMP LOOP      /REPEAT CYCLE
                      /END OF LOOP
```

Executive of this program loop takes 25.5 microseconds plus the conversion time, which is 35 microseconds maximum. Therefore, the worst case conditions for this routine require a 60.5-microsecond execution time and a minimum conversion rate of 16.5 kc.

# CHAPTER 7

# DIGITAL-TO-ANALOG CONVERTER TYPE AA01A

The general purpose Digital-to-Analog Converter Type AA01A converts 12-bit binary computer output numbers to analog voltages. The basic option consists of three channels, each containing a 12-bit digital buffer register and a digital-to-analog converter (DAC). Digital input to all three registers is provided, in common, by one 12-bit input channel which receives bussed output connections from the PDP-8 accumulator. Appropriate precision voltage reference supplies are provided for the converters.

One IOT microinstruction simultaneously selects a channel and transfers a digital number into the selected register. Each converter operates continuously on the content of the associated register to provide an analog output voltage.

Type AA01A options can be specified in a wide range of basic configurations; e.g., with from one to three channels, with or without output operational amplifiers, and with internally or externally supplied reference voltages. Configurations with double buffer registers in each channel are also available.

Each single-buffered channel of the equipment is operated by a single IOT command. Select codes of 55, 56, and 57 are assigned to the AA01A, making it possible to operate nine single-buffered channels or various configurations of double-buffered channels. A typical instruction for the AA01A is:

LOAD DIGITAL-TO-ANALOG CONVERTER 1 (DAL1)

*Octal Code:* 6551

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the accumulator is loaded into the digital buffer register of channel 1.

*Symbol:* AC $=>$ DAC1

The analog output voltage of a standard converter is from ground to $-9.9976$ volts (other voltages are available in equipment containing output operational amplifiers). All binary input numbers are assumed to be 12 bits in length with negative numbers represented in 2's complement notation. An input of $4000_8$ yields an output of ground potential; an input of $0000_8$ yields an output of $-5$ volts; and an input of $1777_8$ yields an output of $-10$ volts minus the analog value of the least significant digital bit. Output accuracy is $\pm 0.0125\%$ of full scale and resolution is $0.025\%$ of full scale value. Response time, measured directly at the converter output, is 3 microseconds for a full-scale step change to 1 least significant bit accuracy. Maximum buffer register loading rate is 2 megacycles.

# CHAPTER 8
# DISPLAY EQUIPMENT

Cathode-ray tube display equipment available for use with the PDP-8 includes the Oscilloscope Display Type 34D and the Precision Display Type 30N. The Light Pen Type 370 operates with either of these devices.

## Oscilloscope Display Type 34D

Type 34D is a two axis digital-to-analog converter and an intensifying circuit, which provides the Deflection and Intensify signals needed to plot data on an oscilloscope. Coordinate data is loaded into an X buffer (XB) or a Y buffer (YB) from bits 2 through 11 of the accumulator. The binary data in these buffers is converted to a −10 to 0 volt Analog Deflection signal. The 30-volt Intensify signal is connected to the grid of the oscilloscope CRT. The duration of this signal, and hence the intensity of the point displayed, is determined by a 2-bit brightness register (BR). The content of the BR controls timing circuits that establish nominal durations of 1-, 2-, or 4-microsecond for the Intensify signal. The BR is loaded from a number contained in the appropriate IOT instruction. Application of power to the computer or pressing of the START key resets the BR to the maximum brightness. Points can be plotted at approximately a 30-kilocycle rate. The instructions for this display are:

### CLEAR X COORDINATE BUFFER (DCX)

*Octal Code:* 6051

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The X coordinate buffer is cleared in preparation for receiving new X-axis display data.

*Symbol:* 0 => XB

### CLEAR AND LOAD X COORDINATE BUFFER (DXL)

*Octal Code:* 6053

*Event Time:* 1, 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The X cordinate buffer is cleared, then loaded with new X-axis data from bits 2 through 11 of the AC.

*Symbol:* 0 => XB
AC2-11 => XB

## CLEAR Y COORDINATE BUFFER (DCY)

*Octal Code:* 6061

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The Y coordinate buffer is cleared in preparation for receiving new Y-axis display data.

*Symbol:* 0 => YB

## CLEAR AND LOAD Y COORDINATE BUFFER (DYL)

*Octal Code:* 6063

*Event Time:* 1, 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The Y coordinate buffer is cleared then loaded with new Y-axis data from bits 2 through 11 of the AC.

*Symbol:* 0 => YB
AC 2-11 => YB

## INTENSIFY (DIX)

*Octal Code:* 6054

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Intensify the point defined by the content of the X and Y coordinate buffers. This command can be combined with the DXL command.

*Symbol:* None

## INTENSIFY (DIY)

*Octal Code:* 6064

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Intensify the point defined by the content of the X and Y coordinate buffers. This command is identical to the DIX command except that it can be combined with the DYL command.

*Symbol:* None

## X COORDINATE SEQUENCE (DXS)

*Octal Code:* 6057

*Event Time:* 1, 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* This command executes the combined functions performed by the DXL and DIX commands. The X coordinate buffer is cleared then loaded from the content of AC2 through AC11, then the point defined by the content of the X and Y buffers is intensified.

*Symbol:* 0 => XB
AC 2-11 => XB
then intensify

## Y COORDINATE SEQUENCE (DYS)

*Octal Code:* 6067

*Event Time:* 1, 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* This command executes the combined functions performed by the DYL and DIY commands. The Y coordinate buffer is cleared, then loaded from the content of bits AC2 through 11, then the point defined by the content of the X and Y coordinate buffers is intensified.

*Symbol:* 0 => YB
AC 2-11 => YB
then intensify

## SET BRIGHTNESS CONTROL (DSB)

*Octal Code:* 607X

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The brightness register (BR) is loaded from the content of bits 10 and 11 of the instruction. When the instruction is 6075 the minimum brightness (0.4 microsecond) is set, when 6076 the medium brightness (0.8 microsecond) is set, and when 6077 the maximum brightness (3.0 microseconds) is set.

*Symbol:* MB10-11 => BR

The following program sequence to display a point assumes that the coordinate data is stored in known addresses X and Y.

```
X,
Y,
BEG,  CLA
      TAD X   /LOAD AC WITH X
      DXL     /CLEAR AND LOAD XB
      CLA
      TAD Y   /LOAD AC WITH Y
      DYS     /CLEAR AND LOAD YB, DISPLAY POINT
```

# Precision CRT Display Type 30N

Type 30N functions are similar to those of the Type 34D Oscilloscope Display in plotting points on a self-contained 16-inch cathode ray tube. A 3-bit brightness register is contained in Type 30N to control the duration of the Intensify signal supplied to the CRT. The content of this register specifies the brightness of the point being displayed according to the following scale:

| BR Content | Intensity |
|---|---|
| 3 | brightest |
| 2 | |
| 1 | |
| 0 | average |
| 7 | |
| 6 | |
| 5 | |
| 4 | dimmest |

The BR register is loaded by jam transfer (transfer ones and zeros so that clearing is not required) from the AC by the instruction:

LOAD BRIGHTNESS REGISTER (DLB)

*Octal Code:* 6074

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The brightness register (BR) is loaded by a jam transfer of information contained in bits 9 through 11 of the AC.

*Symbol:* AC 9-11 $=>$ BR

All other instructions and the instruction sequence are similar to those used in the Type 34D.

# Light Pen Type 370

The light pen is a photosensitive device which detects the presence of information displayed on a CRT. If the light pen is held against the face of the CRT at a point displayed, the display flag will be set to a 1. The light pen display flag is connected into the computer instruction skip facility. The commands are:

SKIP ON DISPLAY FLAG (DSF)

*Octal Code:* 6071

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the display flag is sensed, and if it contains a 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Display Flag $=$ 1, then PC $+1$ $=>$ PC

## CLEAR THE DISPLAY FLAG (DCF)

*Octal Code:* 6072

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The display flag is cleared in preparation for sensing another point on the CRT.

*Symbol:* 0 => Display Flag

# CHAPTER 9

# INCREMENTAL PLOTTER AND CONTROL TYPE 350B

Four models of California Computer Products Digital Incremental Recorder can be operated from a DEC Type 350 Increment Plotter Control. Characteristics of the four recorders are:

| CCP Model | Step Size (inches) | Speed (steps/minute) | Paper Width (inches) |
|---|---|---|---|
| 563 | 0.01 or 0.005 | 12,000 | 31 |
| 565 | 0.01 or 0.005 | 18,000 | 12 |

The principles of operation are the same for each of the four models of Digital Incremental Recorders. Bidirectional rotary step motors are employed for both the X and Y axes. Recording is produced by movement of a pen relative to the surface of the graph paper, with each instruction causing an incremental step. X-axis deflection is produced by motion of the drum; Y-axis deflection, by motion of the pen carriage. Instructions are used to raise and lower the pen from the surface of the paper. Each incremental step can be in any one of eight directions through appropriate combinations of the X and Y axis instructions. All recording (discrete points, continuous curves, or symbols) is accomplished by the incremental stepping action of the paper drum and pen carriage. Front panel controls permit single-step or continuous-step manual operation of the drum and carriage, and manual control of the pen solenoid. The recorder and control are connected to the computer program interrupt and instruction skip facility.

Instructions for the recorder and control are:

SKIP ON PLOTTER FLAG (PLSF)

*Octal Code:* 6501

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter flag is sensed, and if it contains a 1 the content of the PC is incremented by one so the next sequential instruction is skipped.

*Symbol:* If Plotter Flag = 1, then PC +1 => PC

## CLEAR PLOTTER FLAG (PLCF)

*Octal Code:* 6502

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter flag is cleared in preparation for issuing a plotter operation command.

*Symbol:* 0 => Plotter Flag

## PEN UP (PLPU)

*Octal Code:* 6504

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter pen is raised from the surface of the paper.

*Symbol:* None

## PEN RIGHT (PLPR)

*Octal Code:* 6511

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter pen is moved to the right in either the raised or lowered position.

*Symbol:* None

## DRUM UP (PLDU)

*Octal Code:* 6512

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter paper drum is moved upward. This command can be combined with the PLPR and PLDD commands.

*Symbol:* None

### DRUM DOWN (PLDD)

*Octal Code:* 6514

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter paper drum is moved downward.

*Symbol:* None

### PEN LEFT (PLPL)

*Octal Code:* 6521

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter pen is moved to the left in either the raised or lowered position.

*Symbol:* None

### DRUM UP (PLUD)

*Octal Code:* 6522

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter paper drum is moved upward. This command is similar to command 6512 except that it can be combined with the PLPL or PLPD commands.

*Symbol:* None

### PEN DOWN (PLPD)

*Octal Code:* 6524

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The plotter pen is lowered to the surface of the paper.

*Symbol:* None

Program sequence must assume that the pen location is known at the start of a routine since there is no means of specifying an absolute pen location in an incremental plotter. Pen location can be preset by the manual controls on the recorder. During a subroutine, the PDP-8 can track the location of the pen on the paper by counting the instructions that increment position of the pen and the drum.

# CHAPTER 10

# CARD READER AND CONTROL TYPE CR01C

The Card Reader and Control Type CR01C reads standard 12-row, 80-column punched cards at a maximum rate of 100 cards per minute. Cards are read by column, beginning with column 1. One select instruction starts the card moving past the read station. Once a card is in motion, all 80 columns are read. Data in a card column is sensed by mechanical star wheels which close an electrical contact when a hole (binary 1) is detected. Column information is read in one of two program selected modes: alphanumeric and binary. In the alphanumeric mode the 12 information bits in one column are automatically decoded and transferred into the least significant half of the accumulator as a 6-bit Hollerith code. Appendix 2 lists the Hollerith card codes. In the binary mode the 12 bits of a column are transferred directly into the accumulator so that the top row (12) is transferred into AC0 and the bottom row (9) is transferred into AC11. A punched hole is interpreted as a binary 1 and no hole is interpreted as a binary 0.

Three program flags indicate card reader conditions to the computer. The data ready flag rises and requests a program interrupt when a column of information is ready to be transferred into the AC. A read alphanumeric or read binary command must be issued within 1.5 milliseconds after the data ready flag rises to prevent data loss. The card done flag rises and requests a program interrupt when the card leaves the read station. A new select command must be issued within 25 milliseconds after the card done flag rises to keep the reader operating at maximum speed. Sensing of this flag can eliminate the need for counting columns, or combined with column counting can provide a check for data loss. The reader-not-ready flag can be sensed by a skip command to provide indication of card reader power off, no card in the read station, or that a reader failure has been detected. When this flag is raised the reader cannot be selected and select commands are ignored. The reader-not-ready flag is not connected to the program interrupt facility and cannot be cleared under program control. Manual intervention is required to clear the reader-not-ready flag. Instructions for the CR01C are:

## SKIP ON DATA READY (RCSF)

*Octal Code:* 6631

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the data ready flag is sensed, and if it contains a 1 (indicating that information for one card column is ready to be read) the content of the PC is incremented by one so the next sequential instruction is skipped.

*Symbol:* If Data Ready Flag = 1, then PC + 1 => PC

## READ ALPHANUMERIC (RCRA)

*Octal Code:* 6632

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The 6-bit Hollerith code for the 12 bits of a card column are transferred into bits 6 through 11 of the AC, and the data ready flag is cleared.

*Symbol:* AC6-11 V Hollerith Code => AC6-11
0 => Data Ready Flag

## READ BINARY (RCRB)

*Octal Code:* 6634

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The 12-bit binary code for a card column is transferred directly into the AC, and the data ready flag is cleared. Information from the card column is transferred into the AC so that card row 12 enters AC0, row 11 enters AC1, row 0 enters AC2, . . . . and row 9 enters AC 11.

*Symbol:* AC V Binary Code => AC
0 => Data Ready Flag

## SKIP ON CARD DONE FLAG (RCSP)

*Octal Code:* 6671

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the card done flag is sensed, and if it contains a 1 (indicating that the card has passed the read station) the content of the PC is incremented to skip the next sequential instruction.

*Symbol:* If Card Done Flag = 1, then PC + 1 => PC

## SELECT CARD READER AND SKIP IF READY (RCSE)

*Octal Code:* 6672

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the reader-not-ready flag is sensed and if it contains a 1 (indicating that the card reader is ready for programmed operation) the PC is incre-

mented to skip the next sequential instruction; a card is started towards the read station from the feed hopper; and the card done flag is cleared. If the reader-not-ready flag contains a 0 (indicating power is off or no card is in the read station) card selection (motion) does not occur and the skip does not occur.

*Symbol:* If Reader-Not-Ready Flag = 1, then PC + 1 => PC
0 => Card Done Flag

## CLEAR CARD DONE FLAG (RCRD)

*Octal Code:* 6674

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The card done flag is cleared. This command allows a program to stop reading at any point in a card deck.

*Symbol:* 0 => Card Done Flag

A logical instruction sequence to read cards is:

```
START,      RCSE            /START CARD MOTION AND SKIP IF READY
            JMP NOT RDY     /JUMP TO SUBROUTINE THAT TYPES OUT
                            /"CARD READER MANUAL INTERVENTION
                            /REQUIRED" OR HALTS
NEXT,       RCSF            /DATA READY?
            JMP .-1         /NO, KEEP WAITING
            RCRA or RCRB    /YES, READ ONE CHARACTER OR ONE
                            /COLUMN
            DCA I STR       /STORE DATA
            RCSD            /END OF CARD?
            JMP NEXT        /NO, READ NEXT COLUMN
            JMP OUT         /YES, JUMP TO SUBROUTINE THAT CHECKS
                            /CARD COUNT OR REPEATS AT START FOR
                            /NEXT CARD
```

No validity or registration checking is performed by the CR01C. A programmed validity check can be made by reading each card column in both the alphanumeric and the binary mode (within the 1.5 millisecond time limitation), then performing a comparison check.

Before commencing a card reading program energize the reader, load the feed hopper with cards, and manually feed the first card to the read station. The function of the manual controls and indicators are as follows (as they appear from right to left on the card reader):

| Control or Indicator | Function |
|---|---|
| ON/OFF switch | Controls the application of primary power to the reader. When power is applied, the reader is ready to respond to operation of the other keys or programmed commands. |
| AUTO/MAN switch | Controls card reading. In the manual position this switch disables the card feed mechanism so that cards must be manually placed on the read table and registered by pressing the REG key. In the automatic position card motion from the feed hopper through the read station is under program control. |
| REG switch | When the AUTO/MAN switch is in the AUTO position the REG key is used to feed the first card to the read station. When the AUTO/MAN switch is in the MAN position the REG key is used to feed a card manually placed on the read table. |
| SKIP switch | This key is not connected on the CR01C and has no effect on equipment operation. |
| CHECK READER indicator | This lamp is not connected on the CR01C. |
| READY indicator | Lights when the reader is energized and cards are present in the feed hopper. The plastic card cover should always be used on top of a deck of cards to assure that the ready switch and indicator are activated. |
| CARD RELEASE pushbutton | When pressed, this pushbutton (adjacent to the read station) releases a card already in the read station. |

# CHAPTER 11

# CARD READER AND CONTROL TYPE 451

The Card Reader and Control Type 451A operates at a rate of 200 cards per minute, and the Type 451B operates at a rate of 800 cards per minute. Cards are read column by column. Column information is read in either alphanumeric or binary mode. The alphanumeric mode converts the 12-bit Hollerith code of one column into the 6-bit binary-coded decimal-code with code validity checking. The binary mode reads a 12-bit column directly into the PDP-8. Approximately one percent of the computer program running time is required to execute a routine that reads the 80 columns of information at the 200 cards per minute rate.

The control of the card reader differs from the control of other input devices, in that the timing of the read-in sequence is dictated by the device. When the command to fetch a card is given, the card reader reads all 80 columns of information in sequence. To read a column, the program must respond to a flag set as each new column is started. The instruction to read the column must come within 2.2 milliseconds of the flag at 200 cards per minute, or must come within 400 microseconds at 800 cards per minute. The commands for either card reader are:

SKIP ON CARD READER FLAG (CRSF)

*Octal Code:* 6632

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the card reader flag is sensed, and if it contains a 1 (indicating that a card column is present for reading) the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Card Reader Flag = 1, then PC + 1 => PC

READ CARD EQUIPMENT STATUS (CERS)

*Octal Code:* 6634

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the card reader flag and status levels are transferred into the content of bits AC6 through AC9. The AC bit assignments are:

AC6 = Flag is set to 1 (the flag rises after reading each of the 80 rows).

AC7 = Card done.

AC6 = Not ready (covers not in place, power is off, START pushbutton has not been pressed, hopper is empty, stacker is full, a card is jammed, a validity check error has been detected, or the read circuit is defective).

AC9 = End of the file (EOF) (hopper is empty and operator has pushed EOF pushbutton).

*Symbol:* Status => AC6-9

## READ CARD READER BUFFER (CRRB)

*Octal Code:* 6671

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the card column buffer (CCB) is transferred into the AC and the card reader flag is cleared. One CRRB command reads either alphanumeric or binary information.

*Symbol:* CCB => AC

## SELECT ALPHANUMERIC (CRSA)

*Octal Code:* 6672

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The card reader alphanumeric mode is selected and a card is started moving past the read heads. Information read into the CCB is in 6-bit alphanumeric form (the Hollerith code representing the decoded 12 row character in one column).

*Symbol:* None

## SELECT BINARY (CRSB)

*Octal Code:* 6674

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Operation:* The card reader alphanumeric mode is selected and a card is started moving past the read heads. Information read into the CCB is in 12-bit binary form.

*Symbols:* None

Upon instruction to read the card reader buffer, the content of the 12-bit CCB is transferred into the AC. In the alphanumeric mode a 6-bit Hollerith code is transferred into AC6 through AC11 and AC0 through AC5 are cleared. In the binary mode the binary content of the 12 bits (or rows) in a card column are transferred into the AC so that row X is read into AC0, row Y into AC1, row 0 into AC2 . . . . and row 9 into AC11. The mode is specified by either the CRSA or CRSB command and can be changed while the card is being read.

# CHAPTER 12
# CARD PUNCH CONTROL TYPE 450

The Card Punch Control Type 450 permits operation of a standard IBM Type 523 Summary Punch with the PDP-8. Punching can occur at a rate of 100 cards per minute. Cards are punched one row at a time at 40-millisecond intervals.

The card punch dictates the timing of a read-out sequence, much as the card reader controls the read-in timing. When a card leaves the hopper, all 12 rows are punched at intervals of 40 milliseconds. Punching time for each row is 24 milliseconds, leaving 16 milliseconds to load the buffer for the next row. A card punch flag indicates that the buffer is ready to be loaded. The commands for the card punch control are:

### SKIP ON CARD PUNCH FLAG (CPSF)

*Octal Code:* 6631

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The card punch flag is sensed, and if it contains a binary 1 (indicating that the punch buffer is available and can be loaded) the content of the PC is incremented by one so the next sequential instruction is skipped.

*Symbol:* If Card Punch Flag = 1, then PC + 1 => PC

### CARD EQUIPMENT READ STATUS (CERS)

*Octal Code:* 6634

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the card punch flag and the status of the Card Punch Error signal in the control are transferred into bits 10 and 11 of the AC, respectively.

*Symbol:* Card Punch Flag => AC10
Card Punch Error => AC11

### CLEAR PUNCH FLAG (CPCF)

*Octal Code:* 6641

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The card punch flag is cleared in preparation for giving a selector punch command.

*Symbol:* 0 => Card Punch Flag

## SELECT CARD PUNCH (CPSE)

*Octal Code:* 6642

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The card punch is selected and a card is transported to the 80-column punch die from the hopper.

*Symbol:* None

## LOAD CARD PUNCH (CPLB)

*Octal Code:* 6644

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the AC is transferred into a portion of the 80-bit card punch row buffer (CPB). Seven CPLB commands are required to fill the CPB.

*Symbol:* AC => CPB

Since 12 bits are transmitted with each IOT instruction, seven IOT instructions must be issued to load the 80-bit row buffer. The first six loading instructions fill the first 72 bits (or columns); the seventh loads the remaining 8 bits of the buffer from AC bits 4 through 11. After the last row of punching is complete, 28 milliseconds are available to select the next card for continuous punching. If the next card is not requested in this interval, the card punch will stop. The maximum rate of the punch is 100 cards per minute in continuous operation. A delay of 1308 milliseconds follows the command to select the first card; a delay of 108 milliseconds separates the punching of cards in continuous operation.

The card punch flag is connected to the program interrupt and to bit 10 of the CERS instruction. Faults occurring in the punch are detected by status bit 11 of the CERS and signify the punch is disabled, the stacker is full, or the hopper is empty.

A program sequence to punch 12 rows of data on a card can be written as follows, assuming the data to be punched in each row is stored in seven consecutive core memory locations beginning in location 100. The program begins in register PNCH.

```
PNCH,   CLA
        CERS            /READ CARD STATUS
        RAR             /ROTATE PUNCH ERROR BIT (AC11)
                        /INTO LINK
        SZL             /PUNCH ERROR?
        JMP CPERR       /YES, JUMP TO PUNCH ERROR SEQUENCE
        CPSE            /NO, SELECT CARD PUNCH
        CLA
        TAD LOC         /INITIALIZE CARD IMAGE
        DAC 10
        TAD RCNT
        DCA TEM1        /INITIALIZE 12 ROW COUNTS
LP1,    CLA
        TAD GPCT        /INITIALIZE 7 GROUPS PER ROW
        DCA TEM2
        CPSF            /SENSE PUNCH LOAD AVAILABILITY
        JMP .-1
LP2,    CLA
        TAD I 10        /7 GROUPS OF 12 BITS PER ROW
        CPLB            /LOAD BUFFER
        ISZ TEM2
        JMP LP2
        ISZ TEM1        /TEST FOR 12 ROWS
        JMP LP1
        HLT             /END PUNCHING OF 1 CARD
LOC,    77              /LOCATION OF CARD IMAGE
RCNT,   -14             /12 ROWS PER CARD
GPCT,   -7              /7 GROUPS PER ROW
TEM1,   0               /ROW COUNTER
TEM2,   0               /GROUP COUNTER
```

# CHAPTER 13

# AUTOMATIC LINE PRINTER AND CONTROL TYPE 645

The line printer can print 300 lines of 120 characters per minute. Each character is selected from a set of 64 available, by a 6-bit binary code (Appendix 2 lists the ASCII character specified for each code). Each 6-bit code is loaded separately into a core storage printing buffer (LPB) from bits 6 through 11 of the AC. The LPB is divided into two 120-character sections. To load one section of the LPB requires 120 load instructions. A print command causes the characters specified by the last-loaded section of the LPB to be printed on one line. As printing of one section of the LPB is in progress, the other section can be reloaded. After the last character in a line is printed, the section of the LPB from which characters were just printed is cleared automatically. The section of the LPB that is loaded and printed is alternated automatically within the printer and is not program specified.

The line printer can load characters into the LPB at a 10-microsecond rate, clears one section of the LPB in 3 to 6 milliseconds, and moves paper at the rate of one line every 18 milliseconds. When transfer of one code into the LPB is completed, the line printer done flag rises to indicate that the printer is ready to receive another code. When printing of the last character of a section of the LPB is completed, the line printer done flag rises and causes a program interrupt to request reloading of that section of the LPB. A line printer error flag rises and causes a program interrupt if the line printer detects an inoperative condition (printer power off, control circuits not reset, paper supply low, etc.).

A 3-bit format register (FR) in the printer is loaded from bits 9 through 11 of the AC during a print command. This register selects one of eight channels of a perforated tape in the printer to control spacing of the paper. The tape moves in synchronism with the paper until a hole is sensed in the selected channel to halt paper advance. A recommended tape has the following characteristics:

| FR Code (Octal) | Paper Spacing | Tape Track |
|---|---|---|
| 0 | 1 line | 2 |
| 1 | 2 lines | 3 |
| 2 | 3 lines | 4 |
| 3 | 6 lines (1/4 page) | 5 |
| 4 | 11 lines (1/2 page) | 6 |
| 5 | 22 lines (3/4 page) | 7 |
| 6 | 33 lines (line feed) | 8 |
| 7 | top of form | 1 |

The IOT microinstructions which command the line printer are:

## SKIP ON LINE PRINTER ERROR (LSE)

*Octal Code:* 6651

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of line printer error flag is sensed, and if it contains a binary 1, indicating that an error has been detected, the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Line Printer Error Flag = 1, then PC + 1 => PC

## CLEAR PRINTER BUFFER (LCB)

*Octal Code:* 6652

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Both sections of the line printer buffer are cleared in preparation for receiving new character information.

*Symbol:* 0 => LPB

## LOAD PRINTER BUFFER (LLB)

*Octal Code:* 6654

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* A section of the printer buffer is loaded from the content of bits 6 through 11 of the AC, then the AC is cleared.

*Symbol:* AC6 – 11 => LPB, then 0 => AC

## SKIP ON LINE PRINTER DONE FLAG (LSD)

*Octal Code:* 6661

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the line printer done flag is sensed and if it contains a binary 1 the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Line Printer Done Flag = 1, then PC + 1 => PC

## CLEAR LINE PRINTER FLAGS (LCF)

*Octal Code:* 6662

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The line printer done and error flags are cleared

*Symbol:* 0 => Line Printer Done Flag
0 => Line Printer Error Flag

## CLEAR FORMAT REGISTER (LPR)

*Octal Code:* 6654

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The line printer format register (FR) is cleared then loaded from the content of bits 9 through 11 of the AC, and the AC is cleared. The line contained in the section of the printer buffer (LPB) loaded last is printed. Paper is advanced in accordance with the selected channel of the format tape if the content of AC8 is a 1. If AC8 is a 0 paper advance is inhibited.

*Symbol:* 0 => FR
AC9 – 11 => FR
0 => AC
The content of half of the LPB is printed
If AC8 = 1, then advance paper according to format tape channel FR

The following routine demonstrates the use of these commands in a sequence which prints an unspecified number of 120-character lines. This sequence assumes that the printer is not in operation, that the paper is manually positioned for the first line of print, and that one-character words are stored in sequential core memory locations beginning at 2000. The PRINT location starts the routine.

```
PRINT,    LCB          /INITIALIZE PRINTER BUFFER
          CLA
          TAD LOC      /LOAD INITIAL CHARACTER ADDRESS
          DCA 10       /STORE IN AUTO-INDEX REGISTER
LRPT,     TAD CNT      /INITIALIZE CHARACTER COUNTER
          DCA TEMP
LOOP,     LSD          /WAIT UNTIL PRINTING BUFFER READY
          JMP LOOP
          LCF          /CLEAR LINE PRINTER FLAG
          TAD I 10     /LOAD AC FROM CURRENT CHARACTER
                       /ADDRESS
          LLB          /LOAD PRINTING BUFFER
          ISZ TEMP     /TEST FOR 120 CHARACTERS LOADED
          JMP LOOP
```

```
        TAD FRM     /LOAD SPACING CONTROL AND
        LPR         /PRINT A LINE
        JMP LRPT    /JUMP TO PRINT ANOTHER LINE
LOC,    1777        /INITIAL CHARACTER ADDRESS -1
CNT,    -170        /CHARACTER COUNTER = 120 DECIMAL
TEMP,   0           /CURRENT CHARACTER ADDRESS
FRM,    10          /SPACING CONTROL AND FORMAT
```

# CHAPTER 14
# SERIAL MAGNETIC DRUM SYSTEM TYPE 251

The Type 251 Serial Magnetic Drum System is a standard option that serves as an auxiliary data storage device. Information in the PDP-8 can be stored (written) in the drum system and retrieved (read) in sectors of 128 computer words. After program initialization, sectors are transferred automatically between the computer core memory and the drum system, transfer of each word being interleaved with the running computer program under control of the computer data break facility. A word is transferred in parallel (12 bits at a time) and is read or written around the surface of the drum serially (one bit at a time). Within the drum system words consist of 12 information bits and a parity bit. Parity bits are generated internally during writing, and are read and checked during reading. Each word is transferred in about 66 microseconds; a sector transfer is completed in 8.2 milliseconds. Average access time is 8.65 milliseconds (17.3 milliseconds maximum). Track and sector format on the drum surface is such that all transfers require the same amount of time, so track and sector are specified together as an 11-bit address for 128 words.

Drum systems are available with 8, 16, 32, 64, 128, 192, or 256 tracks; each track holds 8 sectors of 128 13-bit words. The various drum system capacities are designated by a letter suffix to the system type number as follows: Type 251A, 8K words; 251B, 16K words; 251C, 32K words; 251D, 65K words; 251E, 131K words; 251F, 196K words; and 251G, 262K words.

Indicator lamps on a front panel usually display the content of the four major registers and the status of control flip-flops. The major registers are:

*Drum Core Location Counter (DCL):* A 15-bit register which addresses the next core memory location to or from which a word is to be transferred. As a word is transferred, DCL is incremented by one.

*Drum Address Register (DAR):* An 11-bit register which addresses the drum track and sector which is currently transferring data. The eight most significant bits of the DAR specify the track and the least significant three bits specify a sector on that track. At the completion of a successful sector transfer (error flag is 0) DAR is incremented by one.

*Drum Final Buffer (DFB):* A 12-bit register under control of the data break facility which is a buffer between the memory buffer register and the drum serial buffer. During writing, the DFB holds the next word to be written. During reading, the DFB stores the word just read from the drum until it is transferred to the PDP-8.

*Drum Serial Buffer (DSB):* A 14-bit register which contains a data word and two control bits. It is a serial-to-parallel converter during drum reading, and a parallel-to-serial converter during drum writing. Information is read from the drum into DSB serially and transferred to DFB in parallel. During drum writing, a word is transferred in parallel from DFB into DSB and written serially around the drum.

## Instructions

The commands for the drum system are as follows:

LOAD DRUM CORE LOCATION COUNTER AND READ (DRCR)

*Octal Code:* 6603

*Event Time:* 1, 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The core memory location information in the AC is transferred into the DCL and the drum is prepared to read one sector of information for transfer to the specified core memory location.*

*Symbol:* AC => DCL
1 => Read Control

### LOAD DRUM CORE LOCATION COUNTER AND WRITE (DRCW)

*Octal Code:* 6605

*Event Time:* 1, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The core memory location information in the AC is transferred into the DCL and the drum is prepared to write on one sector the information beginning at the specified core memory address.*

*Symbol:* AC => DCL
1 => Write Control

### CLEAR DRUM FLAGS (DRCF)

*Octal Code:* 6611

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE on computer and COMPLETION FLAG and ERROR FLAG on the drum system become dark.

*Execution Time:* 3.75 microseconds

*Operation:* Both completion flag and error flag are cleared

*Symbol:* 0 => Completion Flag
0 => Error Flag

### LOAD PARITY AND DATA ERROR (DREF)

*Octal Code:* 6612

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of both the parity error and data timing error flip-flops of the drum control is transferred into bits AC0 and AC1, respectively. The command allows the program to evaluate the cause of an error flag setting.

*Symbol:* Parity Error => AC0
Data Timing Error => AC1

*The sector, track, and core memory address are suitably incremented and allow transfer of the next sequential sector without respecifying addresses. The DRCN instruction must be given within 50 microseconds after the completion flag is set to 1 during the previous sector.

## LOAD THE TRACK AND SECTOR (DRTS)

*Octal Code:* 6615

*Event Time:* 1, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Track and sector information in bits 1-11 of the AC is transferred into the DAR, the completion and error flags are cleared, and a transfer (reading or writing) is begun.

*Symbol:* AC1-11 => DAR
0 => Completion Flag
0 => Error Flag

## SKIP ON DRUM ERROR (DRSE)

*Octal Code:* 6621

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The error flag is sampled and if it contains a 0 (indicating no error has been detected) the PC is incremented to skip the next instruction.

*Symbol:* If Error Flag = 0, then PC + 1 => PC

## SKIP ON DRUM COMPLETION (DRSC)

*Octal Code:* 6622

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The completion flag is sampled and if it contains a 1 (indicating a sector transfer is complete) the PC is incremented to skip the next instruction.

*Symbol:* If completion Flag = 1, then PC + 1 => PC

## INITIATE NEXT TRANSFER (DRCN)

*Octal Code:* 6624

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Both the error and completion flags are cleared, then the transfer of the next sector is initiated.

*Symbol:* 0 => Error Flag
0 => Completion Flag
then start transfer

# Programming

Two instructions cause the transfer of a 128-word sector. The first (DRCR or DRCW) specifies the initial core memory location of the transfer and the direction of the transfer (drum-to-core or core-to-drum). The second instruction (DRTS) specifies the track and sector address and initiates the transfer. Transfer of each word is under control of the computer data break facility and completion of a sector transfer is indicated by a completion flag that causes a program interrupt.

The eight most significant bits of a drum address select one of 256 tracks; the three least significant bits select one of eight sectors on the track. A 300-microsecond gap identifies the beginning of a track (sector 0). Even numbered sectors (0, 2, 4, and 6) are recorded consecutively following the 300-microsecond gap. A 50-microsecond gap identifies the beginning of the odd number sectors (sector 1). Odd numbered sectors (1, 3, 5, and 7) are recorded consecutively following the 50-microsecond gap. This format allows the transfer of two consecutively numbered sectors in one drum revolution, provided the continuation instruction (DRCN) is issued in the first 50 microseconds after the drum flag rises to indicate completion of a sector transfer. The program interrupt subroutine can easily determine that the drum system caused an interrupt, check the number of sectors transferred, check for drum errors by sampling the error flag, and issue the continuation instruction in 50 microseconds.

Because the selection of a track read-write head requires 200 microseconds stabilization time, a new track must be specified during the first 200 microseconds of the 300-microsecond gap for continuous transferring. If selected tracks and sectors are consecutive, uninterrupted transferring may be programmed merely by specifying continuation, since the drum system address and the core memory address are automatically incremented. However, if a data timing or parity error occurs, the track and sector number is not advanced and operations stop at the conclusion of a sector transfer. This feature allows the program to sense for error conditions and to locate the track and sector at which transmission fails.

The drum completion flag is set to 1 upon completion of a sector transfer, causing a program interrupt. The flag is cleared either by a clear flag instruction (DRCF) or automatically when one of two transfer instructions (DRTS, DRCN) is given.

The error flag, which should be checked at the completion of each transfer, indicates either of the following conditions:

(1) That a parity error has been detected after reading from drum to core.

(2) That the Break Request signal from the drum system was not answered within the required 66-microsecond period. This condition occurs either because other devices with higher priority are being serviced by the data break facility, or because an instruction requiring longer than 66 microseconds for completion is in progress when the break request is made. In reading from the drum, a data word is incorrect in core memory. In writing on the drum, the next word has not been received from the computer.

The following program examples indicate the operation of the drum system in single and multiple sector transfers.

## SUBROUTINE TO TRANSFER (READ) ONE SECTOR

```
        CLA          /CALLING SEQUENCE
        TAD ADDR     /INITIAL CORE MEMORY ADDRESS
        JMS READ
        0            /TRACK AND SECTOR ADDRESS
        0            /RETURN
READ,   0
        DRCR         /DRCW TO WRITE
        TAD I READ   /LOAD AC WITH TRACK AND SECTOR ADDRESS
        DRTS
        DRSC         /DONE?
        JMP .-1      /NO
        DRSE         /ERRORS?
        JMP ERR      /JUMP TO ERROR CHECK ROUTINE
        ISZ READ
        JMP I READ   /RETURN
```

## SUBROUTINE TO TRANSFER SUCCESSIVE (TWO) SECTORS

```
        CLA          /CALLING SEQUENCE
        TAD ADDR     /INITIAL CORE MEMORY ADDRESS
        JMS READ
        0            /TRACK AND SECTOR ADDRESS
        0            /RETURN
READ,   0
        DRCR         /DRCW TO WRITE
        TAD I READ   /LOAD AC WITH TRACK AND SECTOR ADDRESS
        DRTS
        DRSC         /DONE?
        JMP .-1      /NO
        DRSE         /ERRORS?
        JMP ERR      /JUMP TO ERROR CHECK ROUTINE
        DRCN         /CLEAR FLAGS, CONTINUE TRANSFER
                     /OF NEXT SECTOR
        DRSC
        JMP .-1
        DRSE
        JMP ERR
        ISZ READ
        JMP I READ   /RETURN
```

# CHAPTER 15
# DECTAPE SYSTEMS

The DECtape system is a standard option for the PDP-8 which serves as an auxiliary magnetic tape data storage facility. The DECtape system stores information at fixed positions on magnetic tape as in magnetic disk or drum storage devices, rather than at unknown or variable positions as is the case in conventional magnetic tape systems. This feature allows replacement of blocks of data on tape in a random fashion without disturbing other previously recorded information. In particular, during the writing of information on tape, the system reads format (mark) and timing information from the tape and uses this information to determine the exact position at which to record the information to be written. Similarly, in reading the same mark and timing information is used to locate data to be played back from the tape.

This system has a number of features to improve its reliability and make it exceptionally useful for program updating and program editing applications. These features are: phase or polarity sensed recording on redundant tracks, bidirectional reading and writing, and a simple mechanical mechanism utilizing hydrodynamically lubricated tape guiding (the tape floats on air and does not touch any metal surfaces).

## DECtape Format

DECtape utilizes a 10-track read/write head. Tracks are arranged in five nonadjacent redundant channels: a timing channel, a mark channel, and three information channels. Redundant recording of each character bit on nonadjacent tracks materially reduces bit drop outs and minimizes the effect of skew. Series connection of corresponding track heads within a channel and the use of Manchester phase recording techniques, rather than amplitude sensing techniques, virtually eliminate drop outs.

The timing and mark channels control the timing of operations within the control unit and establish the format of data contained on the information channels. The timing and mark channels are recorded prior to all normal data reading and writing on the information channels. The timing of operations performed by the tape drive and some control functions are determined by the information on the timing channel. Therefore, wide variations in the speed of tape motion do not affect system performance. Information read from the mark channel is used during reading and writing data, to indicate the beginning and end of data blocks and to determine the functions performed by the system in each control mode.

During normal data reading, the control assembles 12-bit computer length words from four successive lines read from the information channels of the tape. During normal data writing, the control disassembles 12-bit words and distributes the bits so they are recorded on four successive lines on the information channels. A mark channel error check circuit assures that one of the permissible marks is read in every six lines on the tape. This 6-line mark channel sensing requires that data be recorded in 12-line segments (12 being the lowest common multiple of 6-line marks and 4-line data words) which correspond to three 12-bit words.

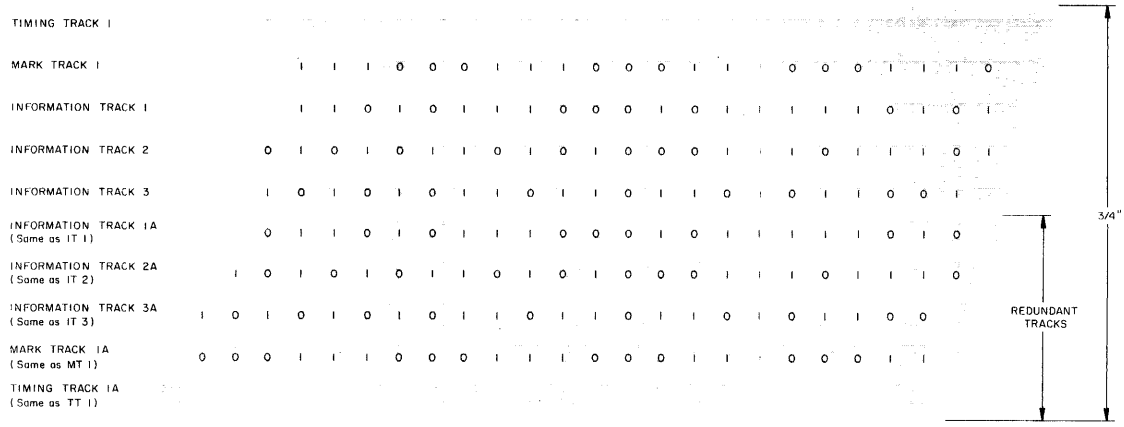



Figure 9 DECtape Track Allocations

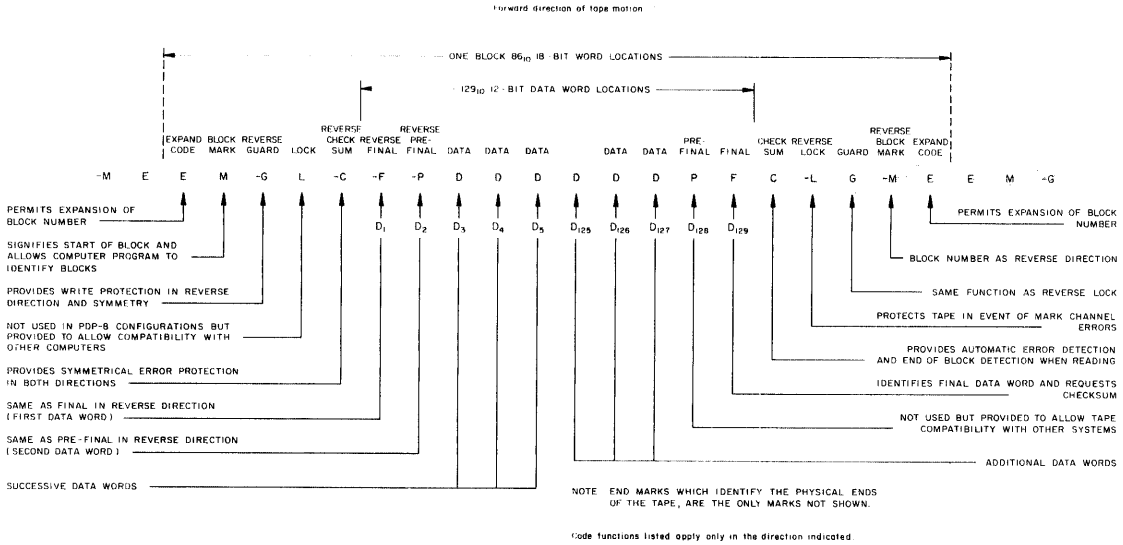



Figure 10 DECtape Mark Channel Format

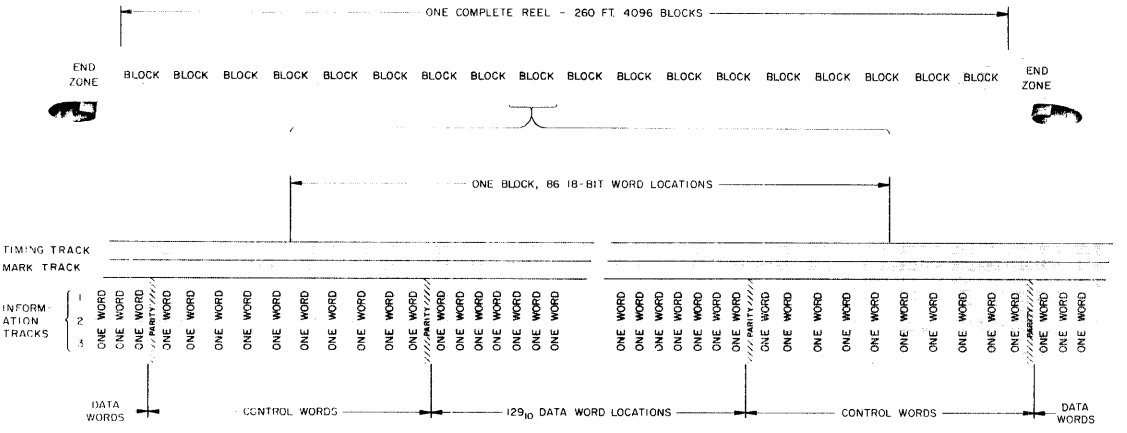



Figure 11 DECtape Control Word and Data Word Assignments

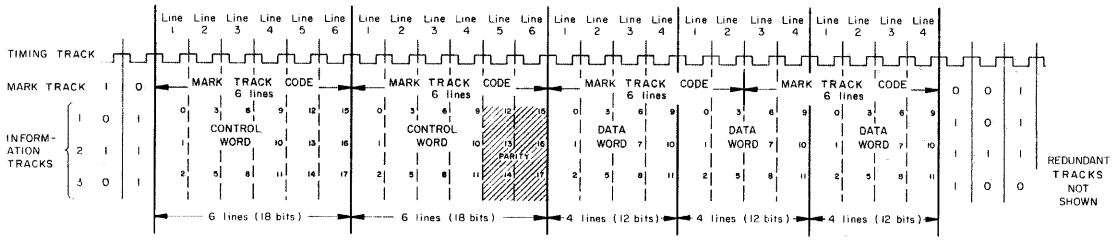



Figure 12 DECtape Format Details

A tape contains a series of data blocks that can be of any length which is a multiple of three 12-bit words. Block length is determined by information on the mark channel. Usually a uniform block length is established over the entire length of a reel of tape by a program which writes mark and timing information at specific locations. The ability to write variable-length blocks is useful for certain data formats. For example, small blocks containing index or tag information can be alternated with large blocks of data. (Software supplied with DECtape allows writing for fixed block lengths only.)

Between the blocks of data are areas called interblock zones. The interblock zones consist of 30 lines on tape before and after a block of data. Each of these 30 lines is divided into five 6-line control words. These 6-line control words allow compatibility between DECtape written on any of DEC's 12-, 18-, or 36-bit computers. As used on the PDP-8, only the last four lines of each control word are used.

Block numbers normally occur in sequence from 1 to N. There is one block numbered 0 and one block N+1. Programs are entered with a statement of the first block number to be used and the total number of blocks to be read or written. The total length of the tape is equivalent to 849,036 lines which can be divided into any number of blocks up to 4096 by prerecording of the mark track. The maximum number of blocks is determined by the following equation in which $N_B$ = number of blocks and $N_W$ = number of words per block ($N_W$ must be divisible by 3).

$$N_B = \frac{212112}{(N_W + 15)} - 2$$

DECtape format is illustrated in Figures 9 through 12.

# DECtape Dual Transport Type 555 and DECtape Control Type 552

The Type 555 Dual DECtape Transport consists of two logically independent tape drives, capable of handling 3.5-inch reels of 0.75-inch magnetic tape. Bits are recorded at a density of 350 ±55 bits per track inch at a speed of over 80 inches per second on the 260-foot length of a reel. Each line on the tape is read or written in approximately 33⅓ microseconds. Simultaneous writing occurs in three channels (three pairs of redundant information tracks), while reading occurs in the mark and timing channels (two pairs of redundant tracks).

The Type 552 DECtape Control operates up to four 555 transports (8 drives) to transfer binary information read from the tape into 12-bit computer words approximately every 133⅓ microseconds. In writing, the control disassembles 12-bit computer words so that they are written at four successive lines on tape. Transfers between the computer and the control always occur in parallel for a 12-bit word. Data transfers use the data break facilities of the computer. As the start and end of each block are detected by the mark track detection circuits, the control raises a DECtape (DT) flag which causes a computer program interrupt. The program interrupt is used by the computer program to determine the block number and when it determines that the forthcoming block is the one selected for a data transfer, it selects the read or write control mode. Each time a word is assembled or the DECtape is ready to receive a word from the computer, the control raises a data flag. This flag is connected to the computer data break facility to signify a break request. Therefore, when the desired block is detected, the data flag causes a data break and initiates a transfer. By using the mark track decoding circuits and data break facility in this manner, computation in the main computer program can continue during tape operations.

*Data Buffer (DB):* This 12-bit register serves as a storage buffer for data to be transferred between DECtape and the computer memory buffer register. During a read operation information sensed from the tape is transferred into the DB from the read/write buffer and is transferred to the computer during a data break cycle. During a write operation the DB received information from the computer and transfers it to the read/write buffer for disassembly and recording on tape. In this manner the DB synchronizes data transfers by allowing transfers between itself and the read/write buffer as a function of the tape timing.

*Read/Write Buffer (R/WB):* This 12-bit register is composed of three 4-bit shift registers. During reading, one bit from each information track is read into a separate segment of the R/WB and shifted right or left as a function of the direction of tape movement. When four tape positions have been read, the content of the R/WB is set into the DB as an assembled 12-bit computer word. During writing, the contents of each segment of the R/WB is shifted serially to the write register (one bit from each of the three segments of the R/WB is transferred into the write register at a time to provide the data to be written at one line) for recording on tape.

*Write Register:* A 3-bit register which is alternately loaded from the R/WB and complemented to write the phase-coded information on tape.

*Select Register:* This 4-bit register is loaded under program control to specify the tape drive selected for operation from the control unit. A single Type 522 DECtape Control can select the drives of four Type 555 Dual DECtape Transports (eight tape drives).

*Motion Register:* This 2-bit register contains a go/stop flip-flop and a forward/reverse flip-flop which control the motion of the selected tape drive. The register is set under program control.

*Longitudinal Parity Buffer (LPB):* This 6-bit register performs a parity check of the information in the three information tracks. The check essentially reads the number of binary zeros in each half of a 12-bit data word and forms a parity bit to be recorded in the checksum control word at the end of the data block. This is effected by setting the information read from two consecutive tape positions into the LPB and then complementing a bit of the LPB if the corresponding bit of the R/WB contains a 0. After reading a block of data the LPB holds a number which indicates the parity of bits 0 and 6, 1 and 7, etc. A 1 in the LPB at this time indicates odd parity and a 0 indicates even parity. When a block of data has been read correctly the LPB contains all binary ones. If the LPB does not contain all ones when a block of data has been read, the parity or mark track error flip-flop is set to 1.

*Memory Address Counter (MAC):* This 12-bit register specifies an address in computer core memory to be used for each word transfer. During program initialization, the starting address of a transfer is set into MAC from the computer accumulator. During the transfer, the address contained in MAC is transferred into the computer memory address register for each data word. The contents of MAC is incremented by 1 at the conclusion of each word transfer so that the transfers occur between successive addresses of computer core memory and tape, regardless of tape direction.

*Window (W):* This 9-bit register serves as a control signal generator for the DECtape system. The mark track data is stored in the W and control signals are generated as a function of the mode of operation in progress and the contents of the W. For example, in the search mode when the W detects a block mark, control signals are generated to raise the DECtape (DT) flag to indicate the presence of a block number in the DB and signals the start of data block to the computer.

*Device Selector (DS):* The device selector is a gating circuit which produces the IOT pulses necessary to initiate operation of the DECtape system and strobe information into the computer.

*Dectape Flag (DT):* This flip-flop serves as an indicator of DECtape system operation to the computer and is connected to the computer program interrupt facility. The function of the DT flag is determined by the control mode in operation at the time, as follows:

a. In the search mode the DT flag rises each time a block mark (block number) is read to indicate the beginning of a new block and to allow programmed determination of the block number which just passed the read/write head.

b. In the read data or write data modes the DT flag rises at the end of each block to indicate the end of a data block. Under these conditions the computer program can sense for this flag to determine when the transfer is complete.

c. In the read all bits or write all bits modes the DT flag rises to indicate completion of each 12-bit word transfer. Since block marks are not observed in these modes, this flag can be used by the computer program to count the number of words transferred as a means of determining tape location.

*Error Flag:* This flag is raised by four error conditions. When the flag rises it initiates a program interrupt to allow the computer interrupt subroutine to determine the condition of the 552 control by means of a read status command. The four error conditions indicated are:

a. *End:* The tape of the selected transport is in the end zone and tape motion is stopped automatically. Under these conditions end is an error if it is not expected by the program in process or is a legitimate signal used to indicate the end of a normal operation (such as rewind) if it is anticipated by the program. If the transport is not selected when the tape enters the end zone this signal is not given, tape motion is not stopped automatically, and the tape can run off the end of the reel.

b. *Timing Error:* The program was not able to keep pace with the tape transfer rate or a new motion or select command was issued before the previous command was completely executed.

c. *Parity or Mark Track Error:* Indicates that during the course of the previous block transfer a data parity error was detected, or one or more bits have been picked up or dropped out from either the timing track or the mark track.

d. *Select Error:* Signifies that a tape transport unit select error has occurred such that more than one transport in the system have been assigned the same select code or that no transport has been assigned the programmed select code.

Therefore, a select error indicates an error by the operator, a timing error is a program error, and a parity or mark channel error indicates an equipment malfunction. Under certain conditions the end may also be an indication of equipment malfunction.

*Data Flag:* This flag is raised each time the DECtape system is ready to transfer a 12-bit word with the computer. When raised, the flag produces a computer data break.

# INSTRUCTIONS

DECtape system commands are microinstructions of the PDP-8 input-output transfer (IOT) instruction, as listed in the following paragraphs:

### LOAD SELECT REGISTER (MMLS)

*Octal Code:* 6751
*Event Time:* 1
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds

*Operation:* The unit select register (USR) is loaded from the content of bits 2 through 5 of the AC, the DECtape flag (DT) is cleared, and a delay is initiated. Loading of the USR involves relay switching which takes approximately 70 milliseconds. When the delay expires the DT flag is set as an indication that loading of the USR is complete and the next DECtape instruction can be given.

*Symbol:* AC2-5 = > USR
1 = > DT when done

### LOAD MOTION REGISTER (MMLM)

*Octal Code:* 6752
*Event Time:* 2
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds

*Operation:* The motion register (MR) is loaded from the content of bits 7 and 8 of the AC, the DECtape flag is cleared, and a 70 millisecond delay is initiated. When the delay expires the DT flag is set, indicating that loading of the MR is completed and the next DECtape instruction can be given.

*Symbol:* AC7-8 = > MR
1 = > DT when done

### LOAD FUNCTION REGISTER (MMLF)

*Octal Code:* 6754
*Event Time:* 3
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds

*Operation:* The function register (FR) is loaded from the content of bits 9 through 11 of the AC, then the AC is cleared. Octal decoding of these three bits establish the following DECtape control modes:

| | |
|---|---|
| 0 = Move | 4 = Write data |
| 1 = Search | 5 = Write all bits |
| 2 = Read data | 6 = Write mark and timing |
| 3 = Read all bits | |

*Symbol:* AC9-11 = > FR,
then 0 = > AC

### SKIP ON DECTAPE FLAG (MMSF)

*Octal Code:* 6761
*Event Time:* 1
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds

*Operation:* The content of the DECtape flag is sensed, and if it contains a binary 1, the content of the PC is incremented by one so the next instruction is skipped.

*Symbol:* If DT = 1, then PC + 1 = > PC

### CLEAR MEMORY ADDRESS COUNTER (MMCC)

*Octal Code:* 6762
*Event Time:* 2
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds
*Operation:* The memory address counter (MAC) is cleared.
*Symbol:* 0 => MAC

### LOAD MEMORY ADDRESS COUNTER (MMLC)

*Octal Code:* 6764
*Event Time:* 3
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds
*Operation:* A transfer of binary ones is performed from the content of the AC into the memory address counter, then the AC is cleared.
*Symbol:* AC V MAC => MAC
then 0 => AC

### SKIP ON ERROR FLAG (MMSC)

*Octal Code:* 6771
*Event Time:* 1
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds
*Operation:* The content of the error flag is sensed, and if it contains a 1 (signifying that an error has been detected) the content of the PC is incremented by one so the next sequential instruction is skipped.
*Symbol:* If Error Flag = 1, then PC + 1 => PC

### CLEAR FLAGS (MMCF)

*Octal Code:* 6772
*Event Time:* 2
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds
*Operation:* Both the DECtape and error flags are cleared.
*Symbol:* 0 => DT, Error Flag

### READ STATUS (MMRS)

*Octal Code:* 6774
*Event Time:* 3
*Indicators:* IOT, FETCH, PAUSE
*Execution Time:* 3.75 microseconds

*Operation:* The condition of the status levels is transferred into bits 0 through 7 of the AC. The AC bit assignments are:

AC0 = DT flag
AC1 = Error flag
AC2 = End (selected tape at end point)
AC3 = Timing error
AC4 = Reverse tape direction
AC5 = Go
AC6 = Parity or mark track error
AC7 = Select error

*Symbol:* Status Levels V AC0-7 => AC0-7

## CONTROL MODES

The seven modes of operation loaded into the function register during the MMLF command are used as follows:

*Move:* Initiates movement of the selected transport tape in either direction. Mark channel errors are inhibited in this mode.

*Search:* As the tape is moved in either direction, sensing a block mark causes both the data flag and the DT flag to rise. The data flag causes a computer data break to deposit the block number in core memory at the address held in MAC. The DT flag initiates a program interrupt to cause the program to jump to a subroutine which is responsible for checking the block numbers by using either the block number stored during this operation or by counting the number of times the DT flag rises.

*Read Data:* A block of data is read in either direction, the data flag rises to cause a data break each time a 12-bit word is to be transferred, and the DT flag is raised to initiate a program interrupt at the end of the data block. The program is responsible for controlling tape motion at the end of a block transfer and must stop motion or change the content of the function register when the DT flag rises. The transport continues reading until taken out of the read data mode.

*Read All Bits:* In this mode of operation the three information channels in data blocks and interblock zones are continuously read and transferred to the computer. This mode is similar to the read data mode except that the DT flag rises each time the data flag rises. The read all bits mode is used to read an unusual tape format which is not compatible with the read data mode. The DT flag is inhibited from causing a program interrupt in this mode

*Write Data:* A block of data is written on tape in either direction, the data flag is raised to effect each transfer, and the DT flag is raised at the end of the block as in the read data mode.

*Write All Bits:* This special mode of operation is used to write information at all positions, disregarding blocks (such as in writing block numbers). The mode is similar to the read all bits mode for writing. The DT flag does not cause a program interrupt in this mode.

*Write Mark and Timing:* Ths mode is used to write on the timing and mark channels to establish or change block length.

## PROGRAMMED OPERATION

Prerecording of a reel of DECtape, prior to its use for data storage, is accomplished in two passes. During the first pass, the timing and mark channels are placed on the tape. During the second pass, forward and reverse block mark numbers, the standard data pattern, and the automatic parity checks are written. Since part of the data word

must be reserved to produce the mark channel, it is impossible to write intelligent data in the information channels at the same time. For this reason, the two passes are required. Prerecording utilizes the write timing and mark channel control mode and a manual switch in the control which permits writing on the timing and mark channels, activates a clock which produces the timing channel recording pattern, and enables flags for program control. Unless both this control mode and switch are used simultaneously, it is physically impossible to write on the mark or timing channels. A red indicator lights on all transports associated with the control when the manual switch is in the "on" position. Under these conditions only, the write register and write amplifier used to write on information channel 0 (bits 0, 3, 6, and 9) is used to write on the mark channel.

Two PDP-8 IOT microinstructions initiate operation of the DECtape system: the first (MMMM) loads the select register, motion register, and function register by means of instruction 6757 (combining MMLS, MMLM, and MMLF) and the second command (MMML is 6766, combining MMCC and MMLC) loads the MAC with the core memory address to be used to store the block number during searching. After initiating operation of the DECtape system, the program should always check for errors immediately by means of the MMSC instruction. This instruction should also be used at the conclusion of each transfer. A program should always start the DECtape system in the search mode to locate the block number selected for a transfer, then when the block number has been located the transfer is accomplished by loading the function register with the read data or write data mode.

In searching, each block number is read by the transport and is transferred to the control. The control raises the DT flag upon receipt of each block number and stores the number in the computer core memory at the address contained in MAC. The computer program, then samples the DT flag and either counts the number of blocks passed or reads the block number from core memory and compares it with the number it is seeking. The results of the data obtained in this way are used to further control the search operation. Upon determining that the forthcoming block is the one selected for a data transfer, the program loads the function register with either the read data or write data mode. Entering another mode discontinues the search mode. The starting address to be used for the first core memory address of the transfer is then set into the MAC by the computer.

When the start of the data position of the block is detected the data flag is raised to initiate a data break each time the DECtape system is ready to transfer a 12-bit word. Therefore, the main computer program continues running but is interrupted approximately every 133⅓ microseconds during a data break for the transfer of a word. Transfers occur between DECtape and successive core memory locations, commencing at the address previously set into MAC. The number of words transferred is determined by the size of the selected tape block. At the conclusion of the block transfer the DT flag is raised and a program interrupt occurs. The interrupt subroutine checks the DECtape error flag to determine the validity of the transfer and either initiates a search for the next information to be transferred or returns to the main program.

During all normal writing transfers, a checksum (the 6-bit exclusive OR of the words in the data block) is computed automatically by the control and is automatically recorded as one of the control words in the interblock zone immediately following the end of the data block. This same checksum is used during reading to determine that the data playback and recognition takes place without error.

Any one of the eight tape drives may be selected for use by the program. After using a particular drive, the program can stop the drive currently being used and select a new drive, or can select another drive while permitting the original selection to continue running. This is a particularly useful feature when rapid searching is desired, since several transports may be used simultaneously. Caution must be exercised however, for although the earlier drive continues to run, no tape end detection or other sensing takes place. Automatic end sensing that stops tape motion occurs in all modes, but only in the selected tape drive.

Whenever either the motion or select code is changed, the program must wait until the DT flag is set to 1 before giving another motion or selection command. In other words, to prevent a timing error all operations of the currently selected drive must be completed before issuing a new select code.

# DECtape Transport Type TU55
# and
# DECtape Control Type TC01

A DECtape system configuration contains up to eight TU55 transports operated from one TC01 control. All data transfers occur between the computer and the control and are effected by the three-cycle data break facility. A 12-bit data buffer in the control synchronizes transfers between the TC01 and the PDP-8 data break facility. Data read from four consecutive lines on tape by the transport are assembled into 12-bit words by a read/write buffer in the control for transfer to the computer. Data loaded into the control from the computer is disassembled by the read/write buffer and supplied to the transport for writing on four lines of tape.

Transfer of command and control signals between the computer and the control is effected by normal IOT instructions. Small registers and control flip-flops in the TC01 are joined to serve as two status registers for the transfer of command and control information with the PDP-8 accumulator. Bit assignments of these registers is indicated in Figure 13 and Figure 14 .

## DECTAPE TRANSPORT TYPE TU55

The TU55 is a bidirectional magnetic-tape transport consisting of a read/write head for recording and playback of information on five channels of the tape. Connections from the read/write head are made directly to the external control which contains the read and write amplifiers.

The logic circuits of the TU55 control tape movement in either direction over the read/write head. Tape drive motor control is exercised completely through the use of solid state switching circuits to provide fast reliable operation. These switching circuits contain silcon controlled rectifiers which are controlled by normal DEC diode and transistor logic circuits. These circuits control the torque of the two motors which transport the tape across the head according to the established function of the device, i.e., go, stop, forward, or reverse. In normal tape movement, full torque is applied to the forward or leading motor and a reduced torque is applied to the reverse or trailing motor to keep proper tension on the tape. Since tape motion is bidirectional, each motor serves as either the leading or trailing drive for the tape, depending upon the forward or reverse control status of the TU55. A positive stop is achieved by an electromagnetic brake

mounted on each motor shaft. When a stop command is given, the trailing motor brake latches to stop tape motion. Enough torque is then applied to the leading motor to take up slack in the tape.

Tape movement can be controlled by commands originating in the computer and applied to the TU55 through the TC01 DECtape Control, or can be controlled by commands generated by manual operation of rocker switches on the front panel of the transport. Manual control is used to mount new reels of tape on the transport, or as a quick maintenance check for proper operation of the control logic in moving the tape.

## DECTAPE CONTROL TYPE TC01

The TC01 DECtape Control operates up to eight TU55 DECtape Transports. Binary information is transferred between the tape and the computer in 12-bit computer words approximately every 133⅓ microseconds. In writing, the control disassembles 12-bit computer words so that they are written at four successive lines on tape. Transfers between the computer and the control always occur in parallel for a 12-bit word. Data transfers use the three-cycle data break facility of the computer. As the start and end of each block of data are detected by the mark track detection circuits, the control raises a DECtape control flag (DTCF) which requests a computer program interrupt. The program interrupt is used by the computer program to determine the block number. When it determines that the forthcoming block is the one selected for a data transfer it establishes the appropriate read or write function. Each time a word is assembled or the DECtape system is ready to receive a word from the computer, the control raises a data flag (DF). This flag is connected to the computer data break facility to request a data break. Therefore, when each 12-bit computer word is assembled, the data flag causes a transfer via the three-cycle data break. By using the mark channel decoding circuits and the data break in this manner, computation in the main computer program can continue during DECtape operations.

Four program flags in the control serve as condition indicators and request originators.

*DECtape Flag (DT):* This flag indicates the active/done status of the current function and is connected to the instruction skip facility.

*Data Flag (DF):* This flag requests a data break to transfer a block number into the computer during a search function, or when a data word transfer is required during a read or write function.

*DECtape Control Flag (DTCF):* This flag, when enabled by a binary 1 in bit 9 of status register A, requests a program interrupt if either the DECtape flag or the error flag is set.

*Error Flag (EF):* Detection of any non-operative condition by the control sets this flag in status register B and stops the selected transport. The error conditions indicated by this flag are:

a. Mark Track Error: This error occurs any time the information read from the mark channel is erroneously decoded.

b. End of Tape: The end zone on either end of the tape is over the read head.

c. Select Error: This error occurs a few microseconds after loading status register A to indicate any one of the following conditions:

   1. Specifying a unit select code which does not correspond to any transport select number, or which is set to multiple transports.
   2. Specifying a write function with the WRITE ENABLED/WRITE LOCK switch in the WRITE LOCK position on the selected transport.

3. Specifying an unused function code (i.e. AC6-8 = 111).
4. Specifying any function except read all with the NORMAL/WRTM/RDMK switch in the RDMK position.
5. Specifying any function except write timing and mark track with the NORMAL/WRTM/RDMK switch in the WRTM position.
6. Specifying the write timing and mark track function with the NORMAL/WRTM/RDMK switch in a position other than WRTM.

d. Parity Error: This error occurs during a read data function if the longitudinal parity over the entire data word, the reverse checksum, and the checksum is not equal to 1.

e. Timing Error: This error indicates a program fault caused by one of the following conditions:

1. A data break did not occur within 66 microseconds of the data break request.
2. The DT flag was not cleared by the program before the control attempted to set it.
3. The read data or write data function was specified while a data block was passing the read head.

## INSTRUCTIONS

Instructions for a TC01/TU55 system are microprogrammed commands of the PDP-8 IOT instruction and are defined as follows:

READ STATUS REGISTER A (DTRA)

*Octal Code:* 6761

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of status register A is loaded into the accumulator by an OR transfer. The AC bit assignments are:

AC0-2 = Transport unit select number
AC3(0) = Forward
AC3(1) = Reverse
AC4(0) = Stop
AC4(1) = Go
AC5(0) = Normal mode
AC5(1) = Continuous mode
AC6-8 = 0 = Move function
AC6-8 = 1 = Search function
AC6-8 = 2 = Read data function
AC6-8 = 3 = Read all function
AC6-8 = 4 = Write data function
AC6-8 = 5 = Write all function
AC6-8 = 6 = Write timing and mark tracks function
AC6-8 = 7 = Unused (causes a select error if issued)
AC9(0) = DECtape control flag and error flag disabled from causing a program interrupt

AC9(1) = DECtape control flag and error flag enabled to cause a program interrupt

*Symbol:* AC0-9 V Status Register A => AC0-9

## CLEAR STATUS REGISTER A (DTCA)

*Octal Code:* 6762

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Status register A is cleared. The DECtape control and error flags are undisturbed.

*Symbol:* 0 => Status Register A

## LOAD STATUS REGISTER A (DTXA)

*Octal Code:* 6764

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The exclusive OR of the content of bits 0 through 9 of the accumulator is loaded into status register A, and bits 10 and 11 of the accumulator are sampled to control clearing of the error and DECtape flags, respectively. Loading status register A from AC0-9 establishes the transport unit select code, motion control, function, and enables or disables the DECtape control flag to request a program interrupt as described in the DTRA instruction. The sampling of AC10 and AC11 is as follows:

AC10(0) = Clear all error flags
AC10(1) = All error flags undisturbed
AC11(0) = Clear DECtape flag
AC11(1) = DECtape flag undisturbed

*Symbol:* AC0-9 V Status Register A => Status Register A

If AC10 = 0, then 0 => EF Flag
If AC11 = 0, then 0 => DT Flag

## SKIP ON FLAGS (DTSF)

*Octal Code:* 6771

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of both the error flags and the DECtape control is sampled, and if any flag contains a binary 1, the content of the program counter is incremented by one to skip the next sequential instruction.

*Symbol:* If EF Flag = 1 V DTCF Flag = 1, then PC + 1 => PC

READ STATUS REGISTUS B (DTRB)

*Octal Code:* 6772

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of status register B is loaded into the accumulator by an OR transfer. The AC bit assignments are:

AC0 = Error flag (Error flag = mark track error V end of tape V select error V parity error V timing error)
AC1 = Mark track error
AC2 = End of tape
AC3 = Select error
AC4 = Parity error
AC5 = Timing error
AC6-8 = Memory field
AC9-10 = Unused
AC11 = DECtape flag

*Symbol:* AC V Status Register B => AC

LOAD STATUS REGISTER B (DTLB)

*Octal Code:* 6774

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The memory field register portion of status register B is loaded by an OR transfer from the content of bits 6 through 9 of the accumulator.

*Symbol:* AC6-8 V Memory Field => Memory Field

## CONTROL MODES

The DECtape system operates in either the normal or continuous mode, as determined by bit 5 of status register A during a DTXA command. Operation in each mode is as follows:

*Normal (NM):* Data transfers and flag settings are controlled by the format of information on the tape.

*Continuous (CM):* Data transfers and flag settings are controlled by a word count (WC) read from core memory during the first cycle of each three-cycle data break; and by the tape format.

## FUNCTIONS

The DECtape system performs one of seven functions, as determined by the octal digit loaded into status register A during a DTXA command. These functions are:

*Move:* Initiates movement of the selected transport tape in either direction. Mark channel decoding is inhibited in this mode except for end of tape.

*Search:* As the tape is moved in either direction, sensing a block mark causes a data transfer of the block number. If the word count overflows (WCO) in either NM or CM, the DT flag is set and causes a program interrupt. After finding the first block number, the CM can be used to avoid all intermediate interrupts between the current and the desired block number. This makes a virtually automatic search possible.

*Read Data:* This function is used to transfer blocks of data into core memory with the transfer controlled by the tape format. In NM the DT flag is set at the end of a block and causes a program interrupt. In CM transfers stop when the word count overflows, the remainder of the block is read for parity checking, and then the DT flag is set.

*Read All:* Read all is used to read tape in an unusual format, since it causes all lines to be read. In NM the DT flag is set at each data transfer. In CM the DT flag is set when WCO occurs. In either case the DT flag causes a program interrupt.

*Write Data:* This function is used to write blocks of data with the transfer controlled by the standard tape format. After WCO occurs, zeros are written in all lines of the tape to the end of the current block. Then the parity checksum for the block is written. The DT flag rises as the in the read data function.

*Write All:* The write all function is used to write an unusual tape format (e. g., block numbers). The DT flag raisings are similar to the read all function.

*Write Timing and Mark Track:* This function is used to write on the timing and mark tracks. This permits blocks to be established or block lengths to be changed. The DT flag raisings are also similar to the read all function. This function is illegal unless a manual switch in the control is on.

## PROGRAMMED OPERATION

Prerecording of a reel of DECtape, prior to its use for data storage, is accomplished in two passes. During the first pass, the timing and mark channels are placed on the tape. During the second pass, forward and reverse block mark numbers, the standard data pattern, and the automatic parity checks are written. These functions are performed by the DECTOG program. Prerecording utilizes the write timing and mark channel function and a manual switch on the control which permits writing on the timing and mark channels, activates a clock which produces the timing channel recording pattern, and enables flags for program control. Unless both this control function and switch are used simultaneously, it is physically impossible to write on the mark or timing channels. A red indicator lamp on the control lights when the manual NORMAL/WRTM/RDMK switch is in the WRTM position. Under these conditions only, the write register and write amplifier used to write on information channel 1 (bits 0, 3, 6, and 9) is used to write on the mark channel. This operation of prerecording need only be performed once for each reel of DECtape.

There are two registers in the TC01 DECtape Control that govern tape operation and provide status information to the operating program. Status register A (see Figure 13) contains three unit selection bits, two motion bits, the continuous mode/normal mode bit, three function bits, and three bits that control the flags. Status register B (see Figure 14) contains the three memory field bits and the error status bits. PDP-8 IOT microinstructions are used to clear, read, and load these registers. In addition, there is an IOT skip instruction to test control status.

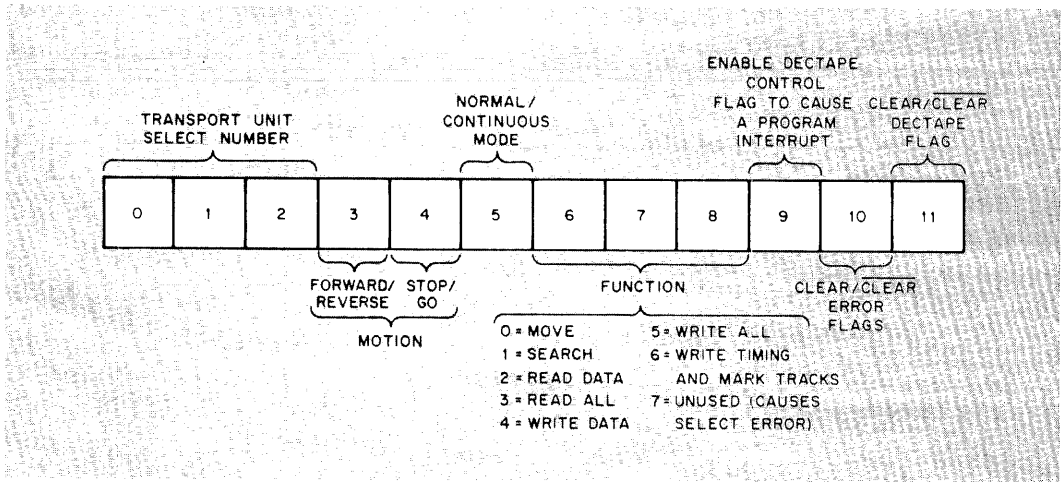



Figure 13 DECtape Status Register A Bit Assignments

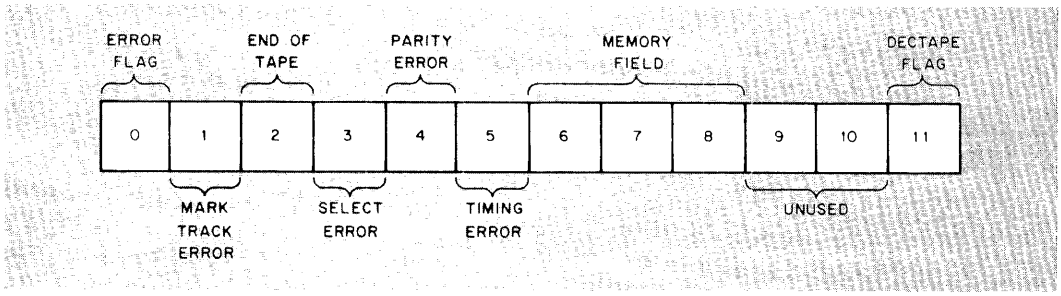



Figure 14 DECtape Status Register B Bit Assignments

Since all data transfers between DECtape and PDP-8 memory are controlled by the data break facility, the program must set the word count (WC) and current address (CA) registers (locations 7754 and 7755 respectively) before a data break. After initiating a DECtape operation, the program should always check for error conditions (a program interrupt would be initiated if the error flag is enabled and if the program interrupt system is enabled). The DECtape system should be started in the search function to locate the block number selected for transfer and then, when the correct block is found, the transfer is accomplished by programmed setting of the WC, CA, and status register A.

When searching, the DECtape control reads only block numbers. These are used by the operating program to locate the correct block number. In NM, the DECtape flag is raised at each block number. In CM, the DECtape flag is raised only after the word count reaches zero. The current address is not incremented during searching and the block number is placed in core memory at the location specified by the content of the CA. Data is transferred to or from the PDP-8 core memory from locations specified by the CA register; which is incremented by one before each transfer.

When the start of the data position of the block is detected, the data flag is raised to initiate a data break request to the data break facility each time the DECtape system is ready to transfer a 12-bit word. Therefore, the main computer program continues running but is interrupted approximately every 133⅓ microseconds for a data break to transfer a word. Transfers occur between DECtape and successive core memory locations specified by the CA. The initial transfer address -1 is stored in the CA by an initializing routine. The number of words transferred is determined only by tape format in NM, or by tape format and the word count in CM. At the conclusion of the data transfer the DT flag is raised and a program interrupt occurs. The interrupt subroutine

checks the DECtape error bits to determine the validity of the transfer and either initiates a search for the next information to be transferred or returns to the main program.

During all normal writing transfers, a checksum (the 6-bit logical equivalence of the words in the data block) is computed automatically by the control and is automatically recorded as one of the control words immediately following the data portion of the block. This same checksum is used during reading to determine that the data playback and recognition take place without error.

Any one of the eight tape transports may be selected for use by the program. After using a particular transport, the program can stop the transport currently being used and select another transport, or can select another transport while permitting the original selection to continue running. This is a particularly useful feature when rapid searching is desired, since several transports may be used simultaneously. Caution must be exercised however, for although the original transport continues to run, no tape end detection or other sensing takes place. Automatic end sensing that stops tape motion occurs in all functions, but only in the selected tape transport.

The following is a list of timing considerations for programmed operations. ($N_S$ = the number of block numbers to be read in the search function and continuous mode, counting through the one causing the WC0. Only the block number causing the WC0 requests a program interrupt. $N_D$ = number of words transferred ÷ the number of words per block. If the remainder ≠ 0, use the next larger whole number. $N_A$ = number of words transferred.)

| Operation | Timing |
|---|---|
| Answer a data break request | Up to 66 microseconds, ±30% |
| Word transfer rate | One 12-bit word every 133 microseconds, ±30% |
| Block transfer rate | One 129-word block every 18.2 milliseconds, ±30% |
| Start time | 200 milliseconds, ±20% |
| Stop time | 150 milliseconds, ±20% |
| Turn around time | 200 milliseconds, ±20% |
| Change function from search to read data for the current block after DT flag from block number | 400 microseconds, ±30% |
| Change function from search to write data for current block after DT flag from block number | 400 microseconds, ±30% |
| Change function from read data to search for the next block after DT flag from transfer completion | 1133 microseconds, ±30% |
| Change function from write data to search for next block after DT flag from transfer completion | 1133 microseconds, ±30% |
| DECtape flag rises | |
| In continuous mode | |
| Move function | Never |
| Search function | ($N_S$) x (18.2 milliseconds, ±30%) |
| Read data function | ($N_D$) x (18.2 milliseconds, ±30%) |
| Read all function | ($N_A$) x (133 microseconds, ±30%) |
| Write data function | ($N_D$) x (18.2 milliseconds, ±30%) |
| Write all function | ($N_A$) x (133 microseconds, ±30%) |
| Write T & M function | ($N_A$) x (133 microseconds, ±30%) |

| Operation | Timing |
|---|---|
| In normal mode | |
| Move function | Never |
| Search function | Every 18.2 milliseconds, ±30% |
| Read data function | Every 18.2 milliseconds, ±30% |
| Read all function | Every 133 microseconds, ±30% |
| Write data function | Every 18.2 milliseconds, ±30% |
| Write all function | Every 133 microseconds, ±30% |
| Write T & M function | Every 133 microseconds, ±30% |

# Software

Three types of programs have been developed as DECtape software for the PDP-8:

a. Subroutines which the programmer may easily incorporate into a program for data storage, logging, data acquisition, data buffering (queing), etc.

b. A library calling system for storing named programs on DECtape and a means of calling them with a minimal size loader.

c. Programs for preformatting tapes controlled by the content of the switch register to write the timing and mark channels, to writ block formats, to exercise the tape and check for errors, and to provide ease of maintenance.

Program development has resulted in a series of subroutines which read or write any number of DECtape blocks, read any number of 129-word blocks as 128 words (one memory page), or search for any block (used by read and write, or to position the tape). These programs are assembled with the user's program and are called by a JMS instruction. The program interrupt is used to detect the setting of the DECtape flag, thus allowing the main program to proceed while the DECtape operation is being completed. A program flag is set when the operation has been completed. Thus, the program effectively allows concurrent operation of several input/output devices along with operation of the DECtape system. These programs occupy two memory pages ($400_8 = 256_{10}$ words).

The library system has the following features: First and perhaps foremost, the system leaves the state of the computer unchanged when it exits. Second, it calls programs by name from the keyboard and allows for expansion of the program file stored on the tape. Finally, it conforms to existing system conventions, namely, that all of memory, except for the last memory page ($7600_8$-$7777_8$), is available to the programmer. This convention ensures that the Binary Loader program (paper tape), and/or future versions of this loader, can reside in memory at all times.

The PDP-8 DECtape library system is loaded by a $17_{10}$-instruction bootstrap routine that starts at address $7600_8$. This loader calls a larger program into the last memory page, whose function is to preserve on the tape, the content of memory from $6000_8$ through $7577_8$, and then load the INDEX program and the directory into those same locations. Since the information in this area of memory has been preserved, it can be restored when operations have been completed. The basic system tape contains the following programs:

a. INDEX: Typing this word causes the names of all programs currently on file to be typed out.

b. UPDATE: Allows the user to add a new program to the files. Update queries the operator about the program's name, its starting address, and its location in core memory.

c. GETSYS: Generates a skeleton library tape on a specified DECtape unit.

d. DELETE: Causes a named file to be deleted from the tape.

Starting with the basic library tape, the user can build a complete file of his active programs and continuously update it. One of the uses of the library tape may be illustrated as follows:

A program is written in PDP-8 FORTRAN that is to be used repeatedly. The programmer may call the FORTRAN compiler from the library tape and with it, compile the program, obtaining the object program. The FORTRAN operating system may then be called from the library tape and used to load the object program. At this time the library program UPDATE is called, the operator defines a new program file (consisting of the FORTRAN operating system and the object program), and adds it to the library tape. As a result, the entire operating program and the object program are now available on the DECtape library tape.

The last group of programs, called DECTOG, is a collection of short routines controlled by the content of the switch register. It provides for the recording of timing and mark channels and permits block formats to be recorded for any block length. Patterns may be written in these blocks and then read and checked. Writing and reading is done in both directions and checked. Specified areas of tape may be "rocked" for specified periods of time. A given reel of tape may thus be thoroughly checked before it is used for data storage. These programs may also be used for maintenance and checkout purposes.

# CHAPTER 16
# AUTOMATIC MAGNETIC TAPE CONTROL TYPE 57A
## Functional Description

This control buffers, compiles, synchronizes, and controls data transfers between up to eight magnetic tape transports and the PDP-8, using program interrupts and data breaks. Each transport requires a small interface circuit for connection to the control. The interface required and the characteristics of the transports that can be connected to the 57A are:

| Transport | Tape Speed (ips) | Densities (bpi) | Interface |
|---|---|---|---|
| DEC Type 50 | 75 | 200/556 | Type 520 |
| DEC Type 545 | 45 | 200/556/800 | Type 521 |
| IBM Model 729II, IV | 75 | 200/556 | Type 552 |
| IBM Model 7330* | 36 | 200/556 | Type 552 |
| IBM Model 729V, VI | 112.5 | 200/556/800 | Type 552 |

The following functions are controlled by various combinations of IOT instructions:

| | |
|---|---|
| Write | Space Backward |
| Write End of File | Rewind |
| Write Blank Tape | Rewind/Unload |
| Read | Write Continuous |
| Read Compare | Read Continuous |
| Space Forward | |

Tape transport motion is governed by one of two control modes: normal, in which tape motion starts upon command and stops automatically at the end of the record; and continuous, in which tape motion starts on command and continues until stopped by the program as a function of synchronizing flags if status conditions appear.

All data transfers are under control of the PDP-8 data break facility; and commands issued during a transfer control, operate, and monitor Type 57A functions by means of the PDP-8 program interrupt facility. Assembled 12-bit PDP-8 data words pass between the computer MB and the control final data buffer register. The core memory address of each word transferred is specified to the computer MA by the control current address register. Use of the program interrupt facility allows the main computer program to continue during long tape operations without running in a loop which waits for tape flags. The program interrupt subroutine for Type 57A loads the AC with numbers, then issues IOT instructions to the control. Specific tape control modes are interpreted from the content of the AC during some IOT instructions. In addition, the current address (CA) register and the word count (WC) registers of the control are loaded from the AC.

Tape functions can be monitored by the program either during or at the end of an operation. They can be altered during operation to a limited degree. The control senses for several types of possible error condition throughout an operation. The results of this sensing can be interrogated by the subroutine at any time.

Two crystal clocks are used to generate one of three character writing rates, depending

*With restrictions

on the density (200, 556, 800) specified by the program. In writing or reading, a composite 12-bit binary word passes between the computer and the control; that is, bits 0 through 5 constitute one tape character, and bits 6 through 11 constitute a second tape character.

In normal operation, six IOT commands initiate reading or writing of one record. When the word count exceeds the number stored in the WC, the transport is stopped and the control is free for another command. In continuous operation, any number of records is written or read without the need for further transport commands except stop.

The following automatic safeguards are inherent in the design of Type 57A:

### END POINT

If the end point is reached during reading or writing, the control ignores the end point and finishes the operation (ample tape is allowed). Beyond the end point tape commands specifying forward direction are illegal and the tape will not respond to such commands. If the end point is passed during spacing, the transport is shut down regardless of word count.

### LOAD POINT

If the load point is reached during back spacing the transport is stopped regardless of word count. At load point, a space back command is legal, and the tape may be unloaded. When the write command is given at load point, the tape is erased 3 inches beyond the load point before writing the first record. After giving a read command at the load point, the read logic is disabled until the load point marker passes the read head, then the read logic is enabled.

### WRITE LOCK RING

Without the write lock ring in the tape reel, writing is illegal and the transport will not respond to a write command.

### FORMAT CONTROL

If the PDP-8 halt command is given during normal reading or read comparing, the tape proceeds to the end of record, and the control shuts down the transport. If a halt is given in continuous reading or read comparing the transport will proceed to end of tape and shut down. If a halt command is given in normal spacing, the transport will proceed to EOR and shut down. If halt is given during continuous spacing, the transport will proceed until WC overflows or until it senses a file marker, load point, or end point, then shut down.

If halt is given during writing in the normal mode, the last word to be transferred is written, the rest of the record is written as zeros, and the transport is shut down. If halt is given during writing in the continuous mode, the record is completed; then zeros are written to the end of the tape. If a WC overflow occurs during a normal read or read compare, the transport proceeds to EOR before shutting down.

## Instructions

The functions of Type 57A Automatic Magnetic Tape Control are controlled by combinations of the following IOT instructions:

## SKIP ON TAPE CONTROL READY (MSCR)

*Octal Code:* 6701

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The Tape Control Ready signal level is sampled, and it is in the binary 1 status, indicating the tape control is free to accept commands, the content of the PC is incremented by one so that the next sequential instruction is skipped.

*Symbol:* If Tape Control Ready = 1, then PC + 1 => PC

## CLEAR AND DISABLE (MCD)

*Octal Code:* 6702

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The job done flag, command register, word count overflow (WCO) flag, and end of record (EOR) flag are cleared. This instruction should be immediately preceded by the two instructions CLA and TAD (4000) to obtain the operation indicated. Both the WCO and EOR flags are connected to the program interrupt facility.

*Symbol:* Inhibit Job Done Flag
0 => Command Register, WCO, EOR

## TAPE SELECT (MTS)

*Octal Code:* 6706

*Event Time:* 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The job done flag is inhibited from causing a program interrupt and clear the WCO and EOR flags are cleared. Then select the transport unit, the mode of parity, and the bit density from the content of the AC. The AC bit assignments are:

AC1(0) = High sense level
AC1(1) = Low sense level
AC2(0) = 200 or 556 bpi density
AC2(1) = 800 or 550 bpi density
AC8(0) = 200 bpi density
AC8(1) = 556 bpi density

| AC2 | AC8 | Density |
|---|---|---|
| 0 | 0 | 200 |
| 0 | 1 | 556 |
| 1 | 0 | 800 |
| 1 | 1 | 556 |

AC7(0) = Even parity (BCD)
AC7(1) = Odd parity (binary)
AC9-11 = These three bits select one of eight tape units, addresses 0 through 7.

*Symbol:* Inhibit Job Done Flag
0 => WCO, EOR
AC1-11 => Select Status

SKIP ON TAPE UNIT READY (MSUR)

*Octal Code:* 6711

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The selected tape transport ready (TTR) status is sampled, and if the transport is ready the content of the PC is incremented by one and the next instruction is skipped. The selected unit must be free before the following MTC command is given.

*Symbol:* If TTR = 1, then PC + 1 => YPC

TERMINATE CONTINUOUS MODE (MNC)

*Octal Code:* 6712

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The CONTINUE flip-flop in the control is cleared to establish the Normal mode of operation, then the AC is cleared. This command should be immediately preceded by CLA and TAD commands to load the AC with the number 4000 to obtain the operation indicated.

*Symbol:* 0 => CONTINUE, AC

LOAD TAPE COMMAND (MTC)

*Octal Code:* 6716

*Event Time:* 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The tape command register is loaded from the content of bits 3 through 6 of the AC. Bit 6 selects the motion mode and bits 3 through 5 are decoded as follows:

AC6(0) = Normal
AC6(1) = Continuous
AC3-5 = 0 = No operation
1 = Rewind
2 = Write
3 = Write end of file (EOF)
4 = Read compare
5 = Read
6 = Space forward
7 = Space backward

*Symbol:* AC3-6 => Tape Command Register

SKIP ON WCO FLAG (MSWF)

*Octal Code:* 6721

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the word count overflow (WCO) flag is sensed, and if it contains a 1 the content of the PC is incremented by one so the next instruction is skipped.

*Symbol:* If WCO = 0, the PC + 1 => PC

DISABLE THE WCO FLAG (MDWF)

*Octal Code:* 6722

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The WCO flag is inhibited from causing a program interrupt.

*Symbol:* None

CLEAR WCO FLAG (MCWF)

*Octal Code:* 6722

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The WCO flag is cleared. This instruction should be immediately preceded by commands that load the number 2000 into the AC to obtain the indicated operation.

*Symbol:* 0 => WCO

ENABLE WCO FLAG (MEWF)

*Octal Code:* 6722

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The WCO flag is enabled to cause a program interrupt upon word count overflow. To obtain this operation the MEWF command must be immediately preceded by a sequence that loads the number 4000 into the AC.

*Symbol:* None

INITIALIZE WCO FLAG (MIWF)

*Octal Code:* 6722

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The WCO flag is initialized. To obtain the operation the MIWF command must be immediately preceded by commands that load the number 6000 into the AC.

*Symbol:* None

## SKIP ON EOR FLAG (MSEF)

*Octal Code:* 6731

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the EOR flag is sensed, and if it contains a 1 the content of the PC is incremented by one so the next instruction is skipped.

*Symbol:* If EOR = 1, then PC + 1 => PC

## DISABLE ERF FLAG (MDEF)

*Octal Code:* 6732

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Operation of the ERF flag is inhibited.

*Symbol:* None

## CLEAR THE ERF FLAG (MCED)

*Octal Code:* 6732

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The ERF flag is cleared to 0 in preparation for returning to the main program from the program interrupt subroutine. This command must be immediately preceded by commands that load the number 2000 into the AC to obtain the indicated operation.

*Symbol:* 0 => ERF

## ENABLE ERF FLAG (MEEF)

*Octal Code:* 6732

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The ERF flag is enabled. This command must be immediately preceded by commands that load the number 4000 into the AC to obtain the indicated operation.

*Symbol:* None

## INITIALIZE ERF FLAG (MIEF)

*Octal Code:* 6732

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The ERF flag is initialized by clearing and enabling. This command must be immediately preceded by commands that load the number 6000 into the AC to obtain the operation indicated.

*Symbol:* None

## READ TAPE STATUS (MTRS)

*Octal Code:* 6734

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The condition of tape status levels is loaded into the AC. This command must be immediately preceded by a command that clears the AC to obtain a valid information transfer. The AC bit assignments are:

AC0 = Data request late
AC1 = Tape parity error
AC2 = Read compare error
AC3 = End of file flag set
AC4 = Write lock ring out
AC5 = Tape at load point
AC6 = Tape at end point
AC7 = Tape near end point (Type 520)
AC7 = Last operation write (Type 521 and 522 interfaces)
AC8 = Tape near load point (Type 520)
AC8 = Write echo (Type 522 interface)
AC8 = B control using transporting (Type 521 interface with multiplex transport)
AC9 = Transport rewinding
AC10 = Tape miss character
AC11 = Job done flag interrupt

*Symbol:* Status Levels => AC0-11

## CLEAR CURRENT COUNT (MCC)

*Octal Code:* 6741

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The current address register (CA) and word count register (WC) are both cleared.

*Symbol:* 0 => CA, WC

READ WORD COUNT (MRWC)

*Octal Code:* 6742

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the AC is transferred into the word count register.

*Symbol:* AC $=>$ WC

READ CURRENT ADDRESS (MRCA)

*Octal Code:* 6744

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the current address register is transferred into the AC. This command does not clear the AC and must be preceded by a command that clears the AC to obtain a valid information transfer.

*Symbol:* CA V AC $=>$ AC

LOAD CURRENT ADDRESS (MCA)

*Octal Code:* 6745

*Event Time:* 1, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The current address register and word count register are both cleared, then the content of the AC is transferred into the current address register.

*Symbol:* 0 $=>$ CA, WC
AC $=>$ CA

# Programming

The following seven control modes are programmed in use of the Type 57A:

## WRITE NORMAL

One of two characters, N words and one or two characters, or N words can be written in BCD mode. When writing BCD, convert all characters ($00_8$) to ($12_8$). The WCO flag is set during the writing of the next to last word in a record. In a one-word transfer only, the WCO flag is set before the data transfer begins. The ERF flag is set when the EOR (check character) is written. Parity is read and compared while writing.

The data request late bit will be set if the PDP-8 does not transfer a new word to or from the control before another data request is given. When a 522 interface is being used, a write echo status appears if the character zero ($00_8$) is written BCD.

## WRITE END OF FILE (EOF)

The end of the marker is written 17, BCD. It is automatically detected during reading or spacing. One instruction, MTC, initiates this operation, carries it out, and stops the transport WCO does not occur. The ERF flag is set when the EOR (check character) is detected. CA and WC are not modified.

## WRITE BLANK TAPE

To write three inches of blank tape, the program gives a write EOF command and then a space backward command. In either case CA and WC are not modified.

## READ NORMAL

One or two characters, N words and one or two characters, or N words can be read in either parity mode. The WCO flag is set during the record when the specified word count is exceeded. The ERF flag is set when the EOR (check character) is detected. Parity errors may be read by examining the appropriate tape status bit.

When reading in BCD mode, convert all $12_8$ to $00_8$. When readin in binary mode and an EOF is detected, the parity error status bit will be set. If while reading, a character does not appear within the allotted time, the miss character status bit will be set.

## READ COMPARE

Words from tape may be compared against consecutive on non-consecutive locations in core memory for equality. An inequality sets the read compare error flag and the CA holds the location of the inequality. Read compare is like read, except that WCO occurs before the last word is compared. The ERF is always set at EOR. Should WCO occur before EOR, the ERF will be set upon comparison of the last word and at EOR.

## SPACE

Spacing forward or backward one record is automatic and does not modify the CA or WC. Spacing N records in either direction can be done on the Continuous mode, and continues until a WCO occurs or EFF is encountered, whichever comes first. If CA is cleared initially, it will contain the record count and may be examined by the program. The program may command stop prematurely with MNC, after which the tape stops as soon as EOR is seen. The parity error flag will be set if a parity error is detected.

## REWIND, REWIND/UNLOAD

Rewind and rewind/unload do not require the use of CA, WC, data interrupt mode, or program interrupt mode. Rewind/unload is selected by specifying rewind and Continuous mode. The transport will not respond to a forward command for 12 milliseconds after the tape has been rewound and stopped at load point.

All operations begin with the program events indicated in the following basic program sequence. When the main program branches to this sequence (having received for example, a high priority data break request from the tape control), the control and transport are interrogated for availability (MSCR, MSUR) and if ready are instructed to carry out the specified task (MTS, MTC). If the task is one of the eight listed in the instruction list under MTC, the MSCR instruction completes the program sequence; if not, the program branches at BEGIN to another routine (write, read, etc.), returning afterwards to WAIT in the basic program.

```
BEGIN,  MSCR            /SKIP IF TAPE CONTROL FREE
        JMP.-1          /TAPE CONTROL NOT FREE, JUMP BACK TO
                        /MSCR INSTRUCTION
        CLA
        TAD  (1A - 1    /LOAD AC WITH INITIAL ADDRESS MINUS ONE
        MCA             /TRANSFER AC TO CA
        CLA
        TAD (- N + 1    /LOAD AC WITH COMPLEMENT OF NUMBER OF
                        /WORDS TO BE TRANSFERRED PLUS ONE
        MRWC            /TRANSFER AC TO WC
        CLA
        TAD (           /LOAD AC WITH SELECTED INFORMATION*
        MTS             /TRANSFER AC TO CONTROL WITH PARITY
                        /DENSITY AND UNIT NUMBER
        MSUR            /SKIP IF TAPE TRANSPORT READY
        JMP.-1          /TRANSPORT NOT READY, JUMP BACK TO
                        /MSUR INSTRUCTION
        MTC             /TRANSFER AC TO CONTROL WITH COMMAND
                        /AND TAPE MOTION MODE
WAIT,   MSCR            /WAIT FOR TAPE FUNCTION TO COMPLETE
        JMP.-1          /TAPE FUNCTION NOT COMPLETE, JUMP
                        /BACK TO MSCR
        HLT             /OPERATION COMPLETION
```

When programming in the interrupt mode, the TCR flag causes an interrupt in the operating program and the flag may be tested by using the MSCR instruction. The TCR flag must be cleared with the MCD command before dismissing the interrupt. WCO and ERF flags must be disabled before dismissing the interrupt with the option of clearing or not clearing the flags.

# CHAPTER 17

# MAGNETIC TAPE SYSTEM TYPE 580

The Magnetic Tape System Type 580 is a semi-automatic data storage system that can be used with the PDP-8. One tape control and one magnetic tape transport constitute the Type 580 system. Data transmission is under program control, while the timing of motion delays, end-of-record delays, write clock pulses, etc., is automatic. Densities are 200 and 556 bits per inch (selected by program), and maximum transfer rate is 25,000 characters per second. Format is compatible with IBM NRZI in either binary or BCD parity mode.

The control contains a 12-bit data buffer, which accumulates the data word in both reading and writing, and a 9-bit command register. All commands, data, and status indications are transferred to or from the computer accumulator.

The system performs the following functions:

Write
Read forward
Read reverse
Space forward (one or N records)
Space reverse (one or N records)
Rewind
Write real time (one word at a time)
Read real time (one word at a time)

These functions are specified by the presence or absence of ones in the accumulator as shown in the following list:

AC1(0) = Sets the SPACE flip-flop
AC3(1) = Sets the GO flip-flop
AC4(1) = Establish write function
AC5(0) = Establish even parity (BCD)
AC5(1) = Establish odd parity (binary)
AC6(1) = Establish read function
AC7(0) = Establish the reverse direction
AC7(1) = Establish the forward direction
AC8(0) = Select density of 200 BPI
AC8(1) = Select density of 556 BPI
AC10(1) = Set rewind
AC11(1) = Set the REAL TIME flip-flop

When the desired function has been encoded in the accumulator, an IOT instruction is given to initiate the pulses that carry out the function. There are nine IOT microinstructions for the Type 580 system, as follows:

## TAPE SYSTEM INITIALIZE FUNCTION AND MOTION (TIFM)

*Octal Code:* 6707

*Event Time:* 1, 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* All tape control registers are cleared, the command register is loaded from bits 1 through 11 of the AC, and motion delays are initiated. The bit assignments of the command register are:

AC1 = Space
AC3 = Go
AC4 = Write
AC5 = Parity mode (0 = even, 1 = odd)
AC6 = Read
AC7 = Direction (0 = reverse, 1 = forward)
AC8 = Density (0 = 200 BPI, 1 = 556 BPI)
AC10 = Rewind
AC11 = Real time

*Symbol:* 0 => All Control Registers
AC1-11 => Command Registers

## TAPE SYSTEM READ (TSRD)

*Octal Code:* 6715

*Event Time:* 1,3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Clear the AC, then load the AC from the content of the data buffer (DB) and clear the data flag.

*Symbol:* 0 => AC
DB => AC
0 => Data Flag

## TAPE SYSTEM WRITE (TSWR)

*Octal Code:* 6716

*Event Time:* 2, 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* Clear the data buffer, then load the data buffer from the content of the AC and clear the data flag.

*Symbol:* 0 => DB
AC => DB
0 => Data Flag

## SKIP ON TAPE SYSTEM DATA FLAG (TSDF)

*Octal Code:* 6721

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the data flag is sampled, and if it contains a 1 the content of the PC is incremented by one so the next instruction is skipped.

*Symbol:* If Data Flag = 1, then PC + 1 => PC

## SKIP ON TAPE SYSTEM END OF RECORD FLAG (TSSR)

*Octal Code:* 6722

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The end of record (EOR) flag is sensed, and if it contains a binary 0 the content of the PC is incremented by one so the next instruction is skipped. The data flag is also sensed, and if it contains a binary 1 the next instruction is skipped.

*Symbol:* If EOR = 0 or if the Data Flag = 1, then PC + 1 => PC

## TAPE SYSTEM STOP DATA TRANSFER (TSST)

*Octal Code:* 6724

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* This instruction is issued following transmission of the last character in a record. It initiates tape shut down procedures such as writing the longitudinal parity bit, end of record mark, and the 0.75-inch inter-record gap. It also clears the SPACE flip-flop, when the correct number of records to be spaced has been reached.

*Symbol:* None

## TAPE SYSTEM READ STATUS (TSRS)

*Octal Code:* 6734

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the status register is transferred into the AC. The bit assignments are:

AC0(1) = Parity error
AC1(1) = Motion delay set
AC2(1) = Transport is ready
AC3(1) = Clock delays set
AC4(1) = End of tape
AC5(1) = Tape at load point

*Symbol:* Status Register => AC0-5

## TAPE SYSTEM WRITE REAL TIME (TWRT)

*Octal Code:* 6731

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* One character is written on tape. This instruction can be used at any frequency and therefore determines the density of information written on tape.

*Symbol:* None

## TAPE SYSTEM CLEAR PROGRAM INTERRUPT (TCPI)

*Octal Code:* 6732

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The status of the program interrupt flag in the control is sampled, and if it is a 1 (indicating that the 580 system caused a program interrupt) the content of the PC is incremented by one so that the next sequential instruction is skipped. This command clears the program interrupt request flag during a space operation and clears the STOP flip-flop.

*Symbol:* If Program Interrupt Request Flag = 1, then PC + 1 => PC
0 => Program Interrupt Request Flag, STOP flip-flop

The following instruction sequence is an example of a routine to write data, and assumes that a previous portion of the program has tested the status of the 580 and that it is ready, etc.

```
/CENTRAL LOOP OF A WRITE DATA ROUTINE

WRT,    CLA                 /GET FIRST WORD
        TAD I AUTO          /VIA AUTO INDEX REGISTER
        TMP TW2             /GO TO A WRITE INSTRUCTION

TW1,    CLA
        TAD I AUTO
        TSDF                /WAIT FOR THE DATA FLAG
        JMP .- 1

TW2,    TSWR                /WRITE
        TSZ CNTR            /COUNT THE NUMBER OF
        JMP TW1             /WORDS TO BE WRITTEN

TW3,    CLA
        TSRS                /READ STATUS
        TSDS                /WAIT FOR LAST
        JMP .- 1            /WORD TO BE WRITTEN
        TSST                /STOP DATA
```

The following instruction sequence indicates the core of a subroutine to read data from the Type 580 system. As such, this routine is unencumbered with initializing and testing operations, and presents the basic commands used with the tape system.

```
/CENTRAL LOOP OF A READ DATA ROUTINE

RED,    CLL CML          /SET THE LINK
        TSDF             /WAIT FOR FIRST CHAR. OR
        JMP .- 1         /WORD TO ENTER BUFFER
        JMP TR2          /GO TO A READ INSTRUCTION

TR1,    TSSR             /SKIP IF END OF RECORD
        JMP .- 1         /OR A DATA FLAG
        TSDF             /SKIP IF DATA FLAG = 1
        JMP TR3          /END OF THE RECORD

TR2,    TSRD             /READ
        SZL              /IF LINK SET
        DCA I AUTO       /STORE IN MEMORY VIA AUTO INDEX
        ISZ CNTR         /COUNT THE WORDS STORED
        JMP TR1
        CLL              /CLEAR LINK TO INHIBIT
        JMP TR1          /STORAGE

TR3,    CLA
        TSRS             /READ STATUS
```

# CHAPTER 18

# DATA COMMUNICATION SYSTEMS TYPE 680

A data communication system consists of a PDP-8 computer with a Data Line Interface Type 681 option, a Serial Line Multiplexer Type 685, and other equipment connected to form a message switching system or to form a data link between serial data transmission equipment and a larger computer. As a message switching system, the 680 system transmits and receives data with up to 128 local or distant Teletype units. As a data link, the 680 system is an economical device for buffering, formatting, and transferring information between a computer and Teletype or other serial processing equipment operating at one or more data speeds. Assuming only minor data handling before transmission to the larger computer, a 680 system can handle up to 128 5-bit Teletype lines at 50 baud. Although the 680 system programming has provision for handling only Teletype lines, programs to pack and unpack massages for other equipment are easily written.

Software for the 680 system is designed to concentrate Teletype data in serial bit format. Although Teletype format is assumed, other data transmission formats that present information in serial format can be used. Subroutines, as presently written, are designed for the 8-bit Teletype code, the 5-bit Teletype code, or a combination of both codes. They also handle mixed speeds on either 8-bit or 5-bit lines with minor changes. Full duplex lines are assumed, but the subroutines operate with half duplex lines, providing the user handles the expected echo.

A Data Communication System Type 680 hardware configuration varies according to the number, type, and distribution of the Teletype units it contains, and upon the use made of the system. Figure 15 shows the basic 680 system configuration, assuming 15 lines: eight local lines, and seven remote lines.

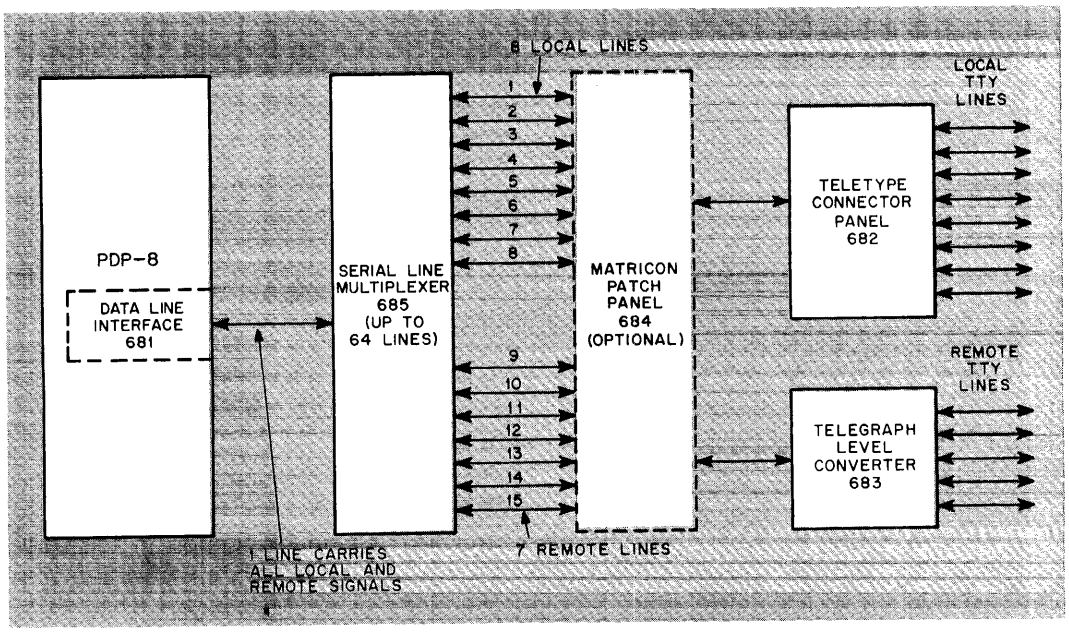



Figure 15 Data Communication System Block Diagram

Teletype signals from remote stations are transmitted and received by a Telegraph Level Converter Type 683. Interface for local units is provided by a Teletype Connector Panel Type 682. Teletype signals for each station run from the 682 or 683 to a Serial Line Multiplexer Type 685. A Matricon Patchboard Type 684 also provides manual selection of channel connections between the 683 and 685. The 685 consists of a multiplexer for Teletype lines and a clock that causes a program interrupt at a rate eight times the line baud frequency. Single line connections are made between the 685 and the Data Line Interface Type 681, and between the 685 and the normal computer interface. The Type 681 option provides an output instruction to transfer Teletype information from the accumulator to the 685 and provides an input instruction to read Teletype information directly into the computer core memory from the 685. All Teletype information transfers occur serially, one bit at a time.

In any serial data transmission system a word consists of an indication that character transmission is about to start, several bits that specify a character code, and an indication that the character is done. Figure 16 shows the format of 11-unit code Teletype words as a typical word format used in serial data transmission. In such a system, the device receiving the word signal must determine the bit sampling time so that information is transferred reliably, even though the digital information signal is severely integrated (pulse rise and fall times increased and pulses rounded) due to transmission path impedance, and even though no synchronization is provided between sending and receiving units or between information on different lines. In addition, jitter (time displacement) of the information signal caused by the mechanical contact nature of the equipment originating the signal, must be considered when determining the strobe time.

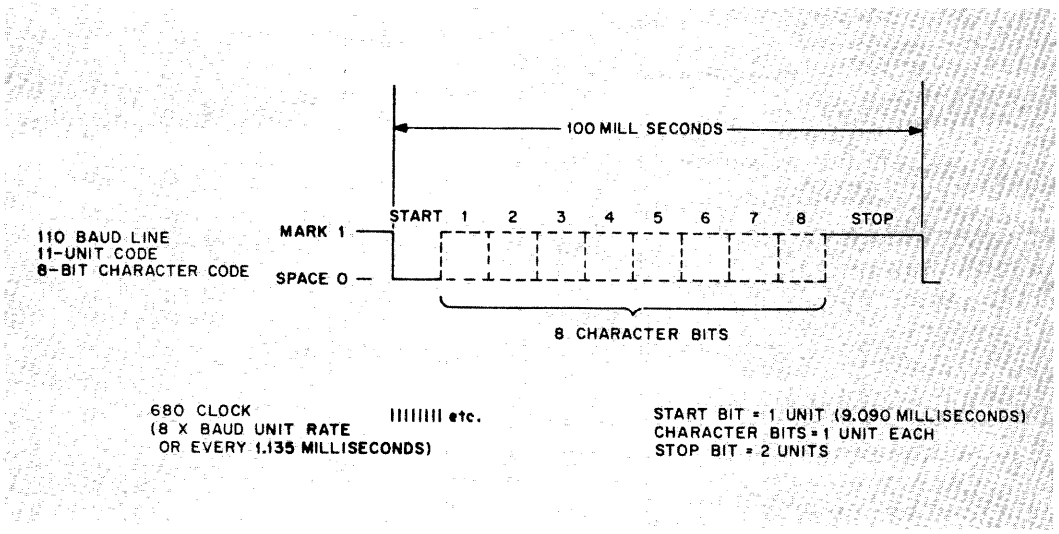



Figure 16 Typical Teletype Line Timing

The Data Communication System Type 680 uses a clock which operates at eight times the bit rate of information on the signal line to determine the sampling time. By counting pulses from this clock, strobe time in receiving and bit timing in transmitting can be controlled within 12.5 percent. Character transmission and reception in the 680 system is controlled by a combination of the hardware and software, providing the most flexibility and economy. This clock in the Serial Line Multiplexer Type 685 requests a program interrupt eight times during each character bit. The program interrupt subroutine counts the clock pulses and strobes a received bit after four clock

pulses have occurred since the line became active, thus assuring that the bit is sampled after the middle of the pulse and within 12.5 percent of the center of the pulse. In like manner, clock pulses are counted by the program interrupt subroutine to transmit a bit after eight clock pulses have occurred.

# Data Line Interface Type 681

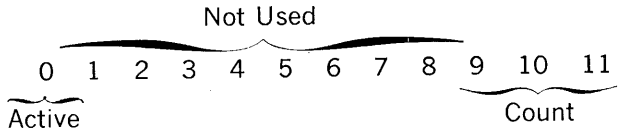
The Type 681 option of the PDP-8 enables use of the computer with a Data Communication System Type 680. The Type 681 option controls and executes transmission and reception of Teletype information between the computer and the Type 680 system. Installation of a Type 681 option in a PDP-8 system adds a Teletype In (TTI) and a Teletype Out (TTO) instruction to the instruction repertoire, and adds two major states to the processor major state generator. The Status (S) and Character (C) states are entered in executing the complex Teletype In instruction.

The TTI and TTO instructions transfer one bit of a Teletype character between the computer and the Serial Line Multiplexer Type 685. These instructions are executed in subroutines entered through the program interrupt subroutine. These subroutines are responsible for determining when a character is completely assembled in the character assembly word (CAW), and for any relocation or translation of assembled characters. Characters are always assembled so that the last bit transmitted shifts into the most significant bit of the CAW and preceding bits are loaded into less significant bits of the CAW, regardless of the Teletype code or transmission path being used.

Teletype In is a complex memory reference instruction which deals with the incoming Teletype line and with two core memory locations. The two locations addressed by the TTI instruction are the next two successive locations following it. These locations contain a line status word (LSW) and a character assembly word (CAW), respectively, so the following sequence is established:

| Address | Content |
|---|---|
| Y | TTI |
| Y + 1 | LSW |
| Y + 2 | CAW |

Bits in the LSW are assigned to record the active/inactive status of the line and serve as a real time clock which determines when line sampling should take place. The format of the LSW is:




The CAW stores partially assembled characters. Individual bits on the incoming line enter the most significant bit (bit 0) and are shifted towards the less significant bit (to the right) during the assembly process.

The TTI instruction is normally executed following a program interrupt caused by the clock in the multiplexer. Figure 17 shows the flow diagram of the TTI instruction.

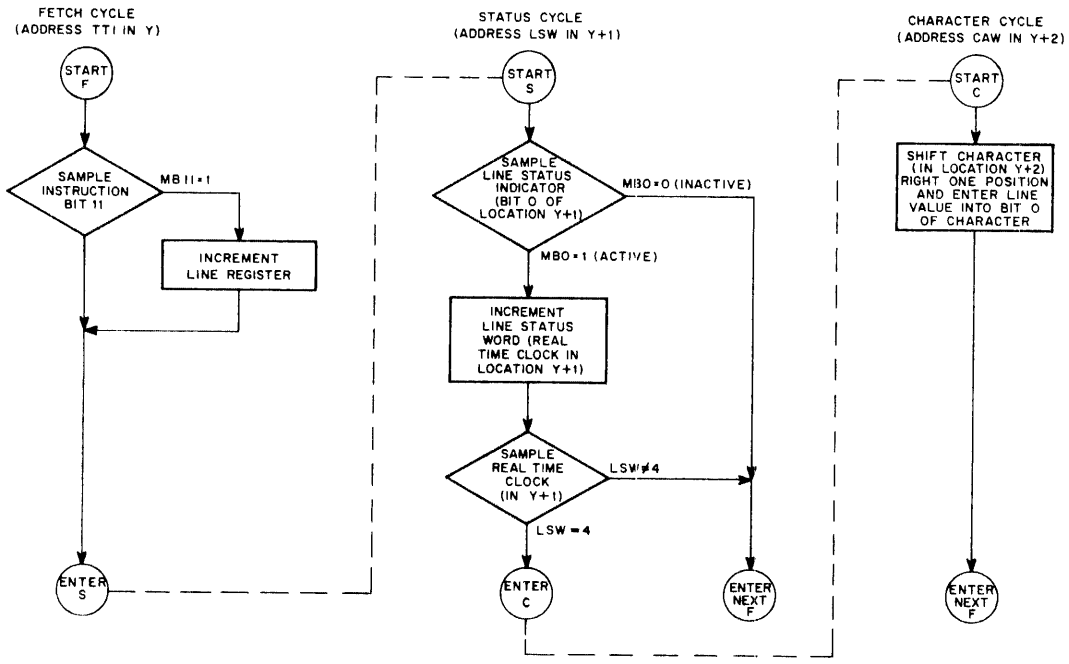



Figure 17 Teletype In Instruction Flow Diagram

Execution of the TTI instruction causes the next location in memory to be read and examined. This location contains the LSW. The first bit examined is the active bit (bit 0). If bit 0 contains a 0, indicating that the line was inactive when last tested, the current content of the line will be set into the active bit. That is, if a start bit is currently being received, the active indicator bit will be set to 1. If no start bit is being received it remains a 0. In either event the character assembly word is skipped over and the next instruction will be executed. Should examination of the active bit indicate that the line is already active, the count portion of the LSW is incremented by one and, unless the resulting count equals 4, the CAW is skipped. When the count becomes equal to 4, indicating that four clock interrupts have been received since the line first became active or that 4/8 of a bit time has elapsed so the center of the bit has been reached and it should be sampled. Thus the character assembly word is read, its content is shifted right one position, and the bit presently being received on the line is set into the leftmost position of the CAW. After the first bit has been received eight clock periods occur before the count is again equal to 4 and thus each bit in the serial train is sampled within 12.5% of the center of the bit.

The Teletype Out instruction affects only the content of the accumulator and the outgoing line. It is executed during a single memory cycle. It shifts the content of the accumulator one position to the right and transmits the least significant bit on the outgoing line. Since a bit is transmitted every time this instruction is executed it should be programmed to occur only after eight clock interrupts have been received since the last output.

These instructions are used only in subroutines and are not used in the main program. The following explanation of their use is for description only.

The Teletype In command brings the bits coming over the line into memory and assembles the bits into one Teletype character. The TTI command uses three memory locations as follows:

```
TTI
0          /status and counter word (LSW)
2000       /character assembly word (for 8-bit code) (CAW)
```

The program then returns to TTI + 3

The character assembly word is preset so that 1 appears in bit 11 when the entire character, including one stop bit, has been shifted in. The subroutines, finding a 1 in bit 11, assume that an entire character has been read, place the character in its own internal buffer together with the line number it came from, and reinitializes the TTI command by resetting the line status word to 0 and the character assembly word to the proper number. At each clock pulse the program only checks 1/8th of the lines (1/4 for 5-bit codes) for completion.

Unlike the TTO command, the TTI command is executed for all lines at each clock-produced program interrupt. However, once the incoming character is started (i.e., bit 0 of the LSW = 1) the first bit (the start code) is read at the fourth pulse and each succeeding bit is read at the eighth pulse thereafter, thus guaranteeing that the bit is read at the optimum time.

The Teletype Out command shifts the content of the accumulator right one position, sends the previous content of bit 11 to the Teletype line specified by the line select register, and brings a 0 into bit 0 of the accumulator. The program sequence to transmit a word from core memory to a Teletype unit might be as follows:

```
TAD   CHAR     /GET CHARACTER TO TRANSMIT
TTO            /SHIFT AND TRANSMIT ONE BIT
DCA   CHAR     /SAVE REMAINDER OF CHARACTER
```

This sequence assumes that the line select register has been loaded with the correct line number using commands for the Serial Line Multiplexer Type 685.

# Serial Line Multiplexer Type 685

The 685 is simply a switch which allows the 681 to be connected to any one of 64 Teletype lines. To select a line, the accumulator is set to the number of the desired line, and its content is then transferred into the line select register of the 685 by an IOT command. The line select register (LSR) may be loaded at any time with a program-selected address, or can be incremented by a command which may be micropro-grammed with the TTI or TTO instructions to scan all lines in numbered sequence. Incrementing is used for high-speed sequential scans. This unit also contains a flip-flop for each outgoing line. This flip-flop is set or cleared by a TTO instruction and holds the line in the proper state until the next TTO instruction is executed.

# Instructions

All instructions for the 680 system contain an operation code of 6, indicating that they are IOT commands. Commands which are associated with the 681 transfer one bit of a character with the computer and have a select code of 40. These commands are functionally memory reference instructions used to perform an input/output transfer operation. Commands associated with the line select register of the 685 use select codes 40 and 41. Commands associated with the clocks of the 685 use select code 42 through 45. All commands using select codes of 41 and 42 use the IOP pulses and are true IOT instructions. The instructions for the 680 system are:

## TELETYPE INCREMENT (TTINCR)

*Octal Code:* 6401

*Event Time:* Not applicable in the normal sense of IOT event times. However it can be considered event time 1, since it is executed before all other operations in the TTI or TTO commands with which it can be combined.

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds when performed individually, or equal to the execution time of other commands when microprogrammed.

*Operation:* The content of the line select register (LSR) in the Serial Line Multiplexer is incremented by one to address the next sequentially numbered line unit. This operation occurs at T1 time of the Fetch cycle.

*Symbol:* LSR + 1 -> LSR

## TELETYPE IN (TTI)

*Octal Code:* 6402

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 3.0 or 4.5 microseconds

*Operation:* Three core memory locations are required by the TTI instruction. The first location contains the TTI instruction, and the two succeeding locations contain a line status word (LSW) and a character assembly word (CAW), respectively. Bit 0 of the LSW records the active/inactive status of the selected Teletype line, and bits 9 through 11 of the LSW serve as a real time clock to determine the bit assembly time for the CAW. Both of these words should be cleared prior to the first use of the TTI instruction in a subroutine. The TTI instruction checks the status of the selected line and the number in the real time clock. If the line is active and the clock indicates the center of a bit has passed, one bit of the Teletype line is shifted into the CAW.

The TTI instruction is executed in two or three computer cyles. The first cycle is in the Fetch state to read the instruction from core memory and to establish the next sequential core memory location as the address to be read during the next cycle. By placing a 1 in bit 11, this instruction can be microprogrammed to increment the content of the flip-flop line register of the Serial Line Multiplexer Type 685 during the Fetch cycle.

The second cycle is a Status state in which the LSW is read, the active/inactive status of the line is checked, the timing of the current bit is checked, and (based on these conditions) the inactive status of the line is recorded in MB0 and the program advances to the next instruction, the real time clock count is incremented in the LSW and the program advances to the next instruction, or the real time clock count is incremented and the third cycle is initiated.

The active/inactive status of the Teletype line is checked by sampling the condition of bit 0 of the LSW. If MB0(0), indicating that the line is inactive (not transmitting a character) the LSW is shifted one position to the right in the MB, and the complement of the Teletype line is set into MB0. Therefore, if the line is now active, a 1 is set into MB0 and will be read during the Status cycle of the next TTI instruction. The program count is then incremented by one to skip over the CAW, the LSW is restored to core memory, the MB is cleared, and (providing no break request had been received) the Fetch state is entered to fetch the next instruction.

If the MB0(1) at the beginning of the Status cycle, the LSW is incremented by one to advance the real time clock and the LSW number is sampled. If LSW≠3 it is too early to sample the active line so the program count is incremented to skip over the CAW, the LSW is restored to core memory, the MB is cleared, and the program advances to the Fetch state for the next instruction. If LSW = 4 after incrementation, the LSW is rewritten in memory and the major state generator (MSG) is set to the Character state to strobe the line into the CAW during the next cycle.

The third cycle is a Character state in which the CAW is read into the MB from core memory, the character is shifted right one position with the line bit being shifted into MB0, then the CAW is rewritten in memory. The program then advances to the Fetch state for the next instruction.

*Symbol:* Status state
If MB0(0), then line shifted into LSW and F=> MSG for next instruction.
If MB0(1), and MB≠3, then LSW + 1 => LSW and F => MSG for next instruction.
If MB0(1) and MB = 3, then LSW + 1 => LSW and C => MSG to continue TTI instruction in next cycle.

Character state
Line shifted into CAW and F = > MSG for next instruction.

## TELETYPE OUT (TTO)

*Octal Code:* 6404

*Event Time:* Not applicable

*Indicators:* IOT, FETCH

*Execution Time:* 1.5 microseconds

*Operation:* This instruction must be preceded by a command sequence (such as CLA and TAD) that loads the AC with the character to be (or being) transferred to the external Teletype equipment. The TTO instruction clears the L, shifts the content of the AC and the L one position to the right, then transfers the bit contained in AC11 to the selected Teletype line.

*Symbol:* 0 => L
L => AC0 and ACj => ACj + 1, then
AC11 => Selected Line

## CLEAR LINE SELECT REGISTER (TTCL)

*Octal Code:* 6411

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The line select register is cleared, so line 0 is addressed.

*Symbol:* 0 => LSR

## LOAD LINE SELECT REGISTER (TTSL)

*Octal Code:* 6412

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The line select register is set by an OR transfer from the content of bits 5 through 11 of the accumulator, then the accumulator is cleared.

*Symbol:* AC5-11 V LSR => LSR, then
0 => AC

## READ LINE SELECT REGISTER (TTRL)

*Octal Code:* 6414

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of the line select register is loaded into bits 5 through 11 of the accumulator by an OR transfer.

*Symbol:* LSR V AC5-11 => AC5-11

## SKIP ON CLOCK 1 FLAG (TTSKP)

*Octal Code:* 6421

*Event Time:* 1

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The content of clock 1 flag of the Serial Line Multiplexer is sampled, and if it contains a 1 (indicating that a clock pulse has occurred and the flag has been enabled to request a program interrupt) the content of the program counter is incremented by 1 to skip the next sequential instruction. If the skip occurs clock 1 caused a program interrupt if the interrupt system was enabled when the clock pulse occurred.

*Symbol:* If Clock 1 Flag = 1, then PC + 1 => PC

## TURN ON CLOCK 1 (TTXON)

*Octal Code:* 6422

*Event Time:* 2

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The CLOCK 1 ENABLE flip-flop is set and the clock 1 flag is cleared. When the CLOCK 1 ENABLE flip-flop is set the next clock pulse sets the clock 1 flag and requests a program interrupt.

*Symbol:* 1 => Clock 1 Enable
0 => Clock 1 Flag

TURN OFF CLOCK 1 (TTXOFF)

*Octal Code:* 6424

*Event Time:* 3

*Indicators:* IOT, FETCH, PAUSE

*Execution Time:* 3.75 microseconds

*Operation:* The CLOCK 1 ENABLE flip-flop is cleared and the clock 1 flag is cleared. When the CLOCK 1 ENABLE flip-flop is cleared the clock 1 flag can not be set by the clock, and can not request a program interrupt or be skipped upon. The clock is unaffected and continues to run, but all operations caused by clock pulses are disabled.

*Symbol:* 0 => Clock 1 Enable
0 => Clock 1 Flag

When the system handles multiple-baud frequencies additional clocks and instructions are provided. Instructions similar to TTSKP, TTXON, and TTXOF use select code 43 for clock 2 and use select code 44 for clock 3.

# Software

Subroutines for the 680 system, as presently coded, occupy $400_8$ core memory locations plus locations for internal buffering of the input and output characters and for thet TTI instructions. In addition, autoindex registers and core memory locations in page 0 are required as specified in the following list:

| 8-Bit | 5-Bit | 5-Bit (2nd speed) | Meaning |
|---|---|---|---|
| TT8BGN | TT5BGN | TT4BGN | Beginning of subroutine |
| T8AX1 | T5AX1 | T4AX1 | Autoindex register |
| T8AX2 | T5AX2 | T4AX2 | Autoindex register |
| T8AX3 | | | |
| T8AX3 | T5AX3 | T4AX3 | Autoindex register |
| | T5AX4 | T4AX4 | Autoindex register (5-bit only) |
| TT8PG0 | TT5PG0 | TT4PG0 | Start of area in page 0 |
| T8O8F2 | T5OBF2 | T4OBF2 | Start of 2nd output buffer (length = N) |
| T81BF | T51BF | T51BF | Start of input buffer (length = 2N) |
| T81N | T51N | T51N | Start of TT1 area (length = 3N + 1) |
| TTCHAR | TTCHAR | TTCHAR | Character area (appears only once) |

The total amount of core memory used by 680 subroutines, including the tags and autoindex registers in page 0, is as follows:

$422_8$ + 7N (for 8-bit)
or
$438_8$ + 7N (for 5-bit)

where N is the number of lines specified to the subroutines. Within limits, the programs can be stored anywhere in the PDP-8 core memory.

If the 5-bit subroutines are being used all of the tags mentioned should substitute 5s for 8s shown. If both 8-bit and 5-bit systems are being used, both sets of subroutines are necessary and all tags and memory requirements must be duplicated for the second system. At present, coding is available for a single 8-bit system and for two different 5-bit systems to allow the programmer to assemble all of the necessary components with a main program at one time.

Percentages of machine time used in the average case for various types of systems are presented in the following list. Any additional features which may be required for the Teletype handling must be added to these times. The formulas for calculating these times are included so that times for systems with an intermediate number of lines or with combinations of lines can be calculated. For combined systems, add the percentages for each component.

| Number of lines | 8-Bit 110 Baud* | 5-Bit 50 Baud** | 5-Bit 75 Baud*** |
|---|---|---|---|
| 32 | 34.1% | 20.0% | 30.0% |
| 64 | 57.7% | 35.1% | 52.7% |
| 96 | 81.3% | 50.3% | 75.5% |
| 128 | 104.9% | 65.5% | 98.3% |

*Formula Used: Where N = the number of lines, the 8-bit subroutines require an average of 8.38N + 119.5 microseconds.

**Formula Used: Where N = the number of lines, the 5-bit subroutines require an average time of 11.85N + 120 microseconds. Clock flags (at 50 baud) occur every 2500 microseconds.

***Formula Used: The percentages for 75 baud are merely 1.5 x 50 baud rate. Clock flags occur every 1667 microseconds.

For further information, refer to DEC Program Library documents DEC-35-S-A and DEC-35-S-B.

# SECTION C

# OPERATION

# CHAPTER 1

# STANDARD PDP-8 OPERATION

## Controls and Indicators

Manual control of the PDP-8 is exercised by means of keys and switches on the operator console. Visual indications of the machine status and the content of major registers and control flip-flops is also given on this console. Indicator lamps light to denote the presence of a binary 1 in specific register bits and in control flip-flops. The function of these controls and indicators is listed in Table 3, and their location is shown in Figure 18. The functions of all controls and indicators of the Model 33 ASR Teletype unit are described in Table 4, as they apply to operation of the computer. The Teletype console is shown in Figure 19.

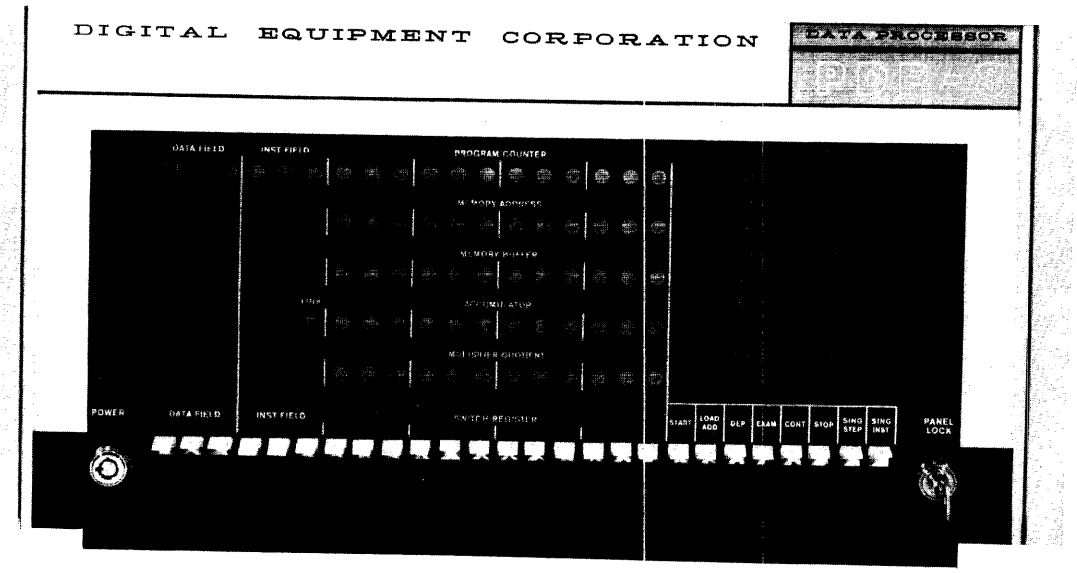



Figure 18 PDP-8 Operator Console

TABLE 3 OPERATOR CONSOLE CONTROLS AND INDICATORS

| Control or Indicator | Function |
|---|---|
| PANEL LOCK switch | With this key-operated switch turned clockwise, all keys and switches except the SWITCH REGISTER switches on the operator console are disabled. In this condition the program can not be disturbed by inadvertent key operation. The program can, however, monitor the content of the SR by execution of the OSR instruction. With this switch turned counterclockwise all operator console keys and switches function normally. |
| POWER switch | In the counterclockwise position this key-operated switch removes primary power from the computer, and in the clockwise position it applies power. |

TABLE 3 OPERATOR CONSOLE CONTROLS AND INDICATORS (continued)

| Control or Indicator | Function |
|---|---|
| START key | Starts the computer program by turning off the program interrupt circuits; clearing the AC, L, MB, and IR; setting the Fetch state, transferring the content of the PC into the MA; and setting the RUN flip-flop. Therefore, the word stored at the address currently held by the PC is taken as the first instruction. |
| LOAD ADDRESS key | Pressing this key sets the content of the SR into the PC, sets the content of the INST FIELD switches into the IF, and sets the content of the DATA FIELD switches into the DF. |
| DEPOSIT key | Lifting this key sets the content of the SR into the MB and core memory at the address specified by the current content of the PC. This operation is performed by setting the Execute state and forcing a DCA instruction. The content of the PC is then incremented by one, to allow storing of information in sequential memory addresses by repeated operation of the DEPOSIT key. |
| EXAMINE key | Pressing this key sets the content of core memory at the address specified by the content of the PC into the MB and AC. This operation is performed by clearing the AC, setting the Execute state, and forcing a TAD instruction. The content of the PC is then incremented by one to allow examination of the content of sequential core memory addresses by repeated operation of the EXAMINE key. |
| CONTINUE key | Pressing this key sets the RUN flip-flop to continue the program in the state and instruction designated by the lighted console indicators, at the address currently specified by the PC. |
| STOP key | Causes the RUN flip-flop to be cleared at the end of the cycle in progress at the time the key is pressed. |
| SINGLE STEP switch | The switch is off in the down position. In the up position the switch causes the RUN flip-flop to be cleared to disable the timing circuits at the end of one cycle of operation. Thereafter, repeated operation of the CONTINUE key steps the program one cycle at a time so that the content of registers can be observed in each state. |

TABLE 3 OPERATOR CONSOLE CONTROLS AND INDICATORS (continued)

| Control or Indicator | Function |
|---|---|
| SINGLE INSTRUCTION switch | The switch is off in the down position. In the up position the switch causes the RUN flip-flop to be cleared at the end of the next instruction execution. When the computer is started by means of the START or CONTINUE key, this switch causes the RUN flip-flop to be cleared at the end of the last cycle of the current instruction. Therefore, repeated operation of the CONTINUE key steps the program one instruction at a time. |
| SWITCH REGISTER switches | Provide a means of manually setting a 12-bit word into the machine. Switches in the up position; corresponds to binary ones, down to zeros. The content of this register is loaded into the PC by the LOAD ADDRESS key or into the MB and core memory by the DEPOSIT key. The content of the SR can be set into the AC under program control by means of the OSR instruction. |
| DATA FIELD indicators and switches* | The indicators denote the content of the data field register (DF) and the switches serve as an extension of the SR to load the DF by means of the LOAD ADDRESS key. The DF determines the core memory field of data storage and retrieval. |
| INST FIELD indicators and switches* | The indicators denote the content of the instruction field register (IF) and the switches serve as an extension of the SR to load the IF by means of the LOAD ADDRESS key. The IF determines the core memory field from which instructions are to be taken. |
| PROGRAM COUNTER indicators | Indicate the content of the PC. When the machine is stopped the content of the PC indicates the core memory address of the first instruction to be executed when the START or CONTINUE key is operated. When the machine is running the content of the PC indicates the core memory address of the next instruction. |
| MEMORY ADDRESS indicators | Indicate the content of the MA. Usually the content of the MA denotes the core memory address of the word currently or previously read or written. After operation of either the DEPOSIT or EXAMINE key, the content of the MA indicates the core memory address at which information was just written or read. |

* Activated only on systems containing the Type 183 Memory Extension Control option.

TABLE 3 OPERATOR CONSOLE CONTROLS AND INDICATORS (continued)

| Control or Indicator | Function |
|---|---|
| MEMORY BUFFER indicators | Indicate the content of the MB. Usually the content of the MB designates the word just read or written at the core memory address held in the MA. |
| ACCUMULATOR indicators | Indicates the content of the AC. |
| LINK indicator | Indicates the content of the L. |
| MULTIPLIER QUOTIENT indicators* | Indicate the content of the multiplier quotient (MQ). The MQ holds the multiplier at the beginning of a multiplication and holds the least significant half of the product at the conclusion. It holds the least significant half of the dividend at the start of a division and at the end holds the quotient. |
| Instruction indicators (AND, TAD, ISZ, DCA, JMS, JMP, IOT, OPR) | Indicate the decoded output of the IR as the instruction currently in progress. |
| FETCH, EXECUTE DEFER, BREAK indicators | Indicate the primary control state of the machine and that the current memory cycle is a Fetch, Execute, Defer or Break cycle, respectively. |
| ION indicator | Indicates the 1 status of the INT. ENABLE flip-flop. When lit, the program in progress can be interrupted by receipt of a Program Interrupt Request signal from an I/O device. |
| PAUSE indicator | Indicates the 1 status of the PAUSE flip-flop when lit. An IOT instruction sets the PAUSE flip-flop at T1 time to initiate operation of the IOP generator and to inhibit advance of the normal timing generator. When IOP generator operation is completed (approximately 2.5 microseconds later), a T2 pulse is generated and the PAUSE flip-flop is cleared to enable advance of the timing generator in synchronism with the basic computer clock. |
| RUN indicator | Indicates the 1 status of the RUN flip-flop. When lit, the internal timing circuits are enabled and the machine performs instructions. |

*Activated only on systems containing the Type 182 Extended Arithmetic Element option.

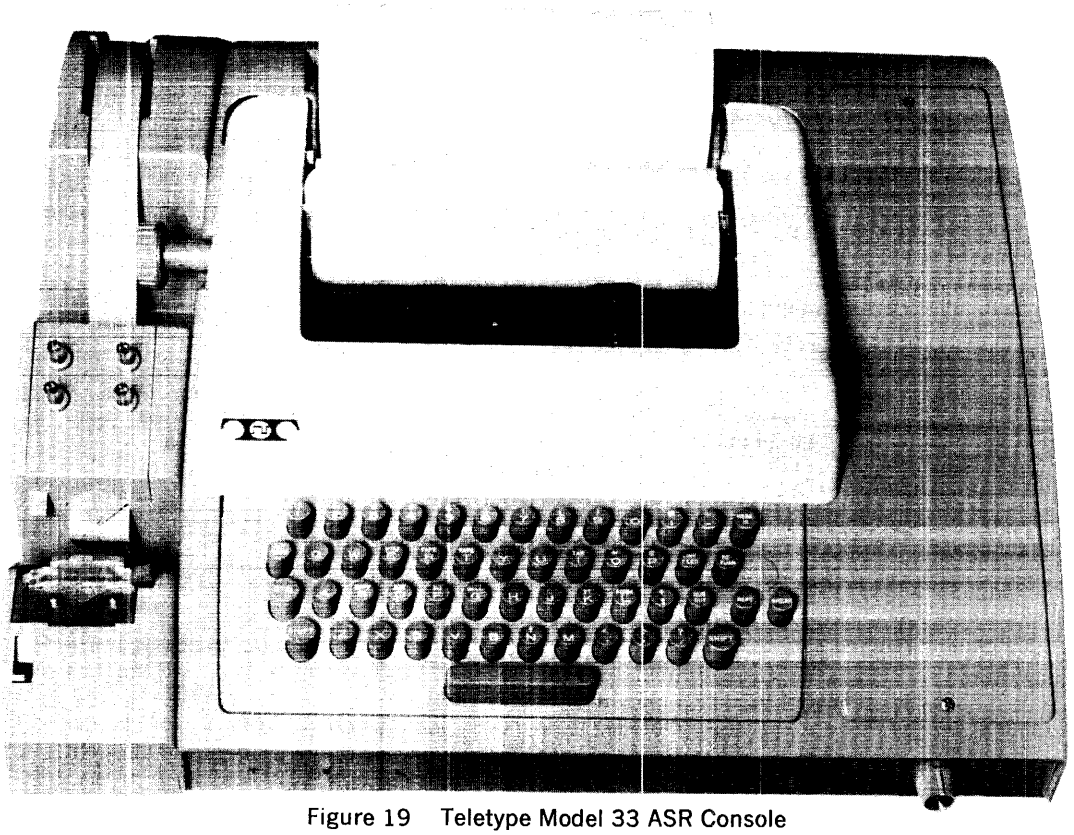


Figure 19 Teletype Model 33 ASR Console

TABLE 4 TELETYPE CONTROLS AND INDICATORS

| Control or Indicator | Function |
|---|---|
| REL. pushbutton | Disengages the tape in the punch to allow tape removal or tape loading. |
| B. SP. pushbutton | Backspaces the tape in the punch by one space, allowing manual correction or rub out of the character just punched. |
| OFF and ON pushbuttons | Control use of the tape punch with operation of the Teletype keyboard/printer. |
| START/STOP/FREE switch | Controls use of the tape reader with operation of the Teletype. In the lower FREE position the reader is disengaged and can be loaded or unloaded. In the center STOP position the reader mechanism is engaged but de-energized. In the upper START position the reader is engaged and operated under program control. |
| Keyboard | Provides a means of printing on paper in use as a typewriter and punching tape when the punch ON pushbutton is pressed, and provides a means of supplying input data to the computer when the LINE/OFF/LOCAL switch is in the LINE position. |
| LINE/OFF/LOCAL switch | Controls application of primary power in the Teletype and controls data connection to the processor. In the LINE position the Teletype is energized and connected as an I/O device of the computer. In the OFF position the Teletype is de-energized. In the LOCAL position the Teletype is energized for off-line operation, and signal connections to the processor are broken. Both line and local use of the Teletype require that the computer be energized through the POWER switch. |

# Operating Procedures

Many means are available for loading and unloading PDP-8 information. The means used are, of course, dependent upon the form of the information, time limitations, and the peripheral equipment connected to the computer. The following procedures are basic to any use of the PDP-8, and although they may be used infrequently as the programming and use of the computer become more sophisticated, they are valuable in preparing the initial programs and learning the function of machine input and output transfers.

## MANUAL DATA STORAGE AND MODIFICATION

Programs and data can be stored or modified manually by means of the facilities on the operator console. Chief use of manual data storage is made to load the readin mode loader program into the computer core memory. The readin mode (RIM) loader is a program used to automatically load programs into PDP-8 from perforated tape in RIM format. This program and the RIM tape format are described in Appendix 6 of this handbook and in Digital Program Library descriptions. The RIM program listed in the Appendix can be used as an exercise in manual data storage. To store data manually in the PDP-8 core memory:

1. Turn the PANEL LOCK switch counterclockwise and turn the POWER switch clockwise.
2. Set the bit switches of the SWITCH REGISTER (SR) to correspond with the address bits of the first word to be stored. Press the LOAD ADDRESS key and observe that the address set by the SR is held in the PC, as designated by lighted PROGRAM COUNTER indicators corresponding to switches in the 1 (up) position and unlighted indicators corresponding to switches in the 0 (down) position.
3. Set the SR to correspond with the data or instruction word to be stored at the address just set into the PC. Lift the DEPOSIT key and observe that the MB, and hence the core memory, hold the word set by the SR.

   Also, observe that the PC has been incremented by one so that additional data can be stored at sequential addresses by repeated SR setting and DEPOSIT key operation.

To check the content of an address in core memory, set the address into the PC as in step 2, then press the EXAMINE key. The content of the address is then designed by the MEMORY BUFFER and ACCUMULATOR indicators. The content of the PC is incremented by one with operation of the EXAMINE key, so the content of sequential addresses can be examined by repeated operation after the original (or starting) address is loaded. The content of any address can be modified by repeating both steps 2 and 3.

## LOADING DATA UNDER PROGRAM CONTROL

Information can be stored or modified in the computer automatically only by enacting programs previously stored in core memory. For example, having the RIM loader stored in core memory allows RIM format tapes to be loaded as follows:

1. Turn the PANEL LOCK switch counterclockwise and turn the POWER switch clockwise.
2. Set the Teletype LINE/OFF/LOCAL switch to the LINE position.

3. Load the tape in the Teletype reader by setting the START/STOP/FREE switch to the FREE position, releasing the cover guard by means of the latch at the right, loading the tape so that the sprocket wheel teeth engage the feed holes in the tape, closing the cover guard, and setting the switch to the STOP position. Tape is loaded in the back of the reader so that it moves toward the front as it is read. Proper positioning of the tape in the reader finds three bit positions being sensed to the left of the sprocket wheel and five bit positions being sensed to the right of the sprocket wheel.

4. Load the starting address of the RIM loader program (not the address of the program to be loaded) into the PC by means of the SR and the LOAD ADDRESS key.

5. Press the computer START key and set the 3-position Teletype reader switch to the START position. The tape will be read automatically.

Automatic storing of the binary loader (BIN) program is performed by means of the RIM loader program as previously described. With the BIN loader stored in core memory, program tapes assembled in the program assembly language (PAL III) binary format can be stored as described in the previous procedure except that the starting address of the BIN loader (usually 7777) is used in step 4. When storing a program in this manner, the computer stops and the AC should contain all zeros if the program is stored properly. If the computer stops with a number other than zero in the AC, a checksum error has been detected. When the program has been stored, it can be initiated by loading the program starting address (usually designated on the leader of the tape) into the PC by means of the SR and LOAD ADDRESS key, then pressing the START key.

## OFF-LINE TELETYPE OPERATION

The Teletype can be used separately from the PDP-8 for typing, punching tape, or duplicating tapes. To use the Teletype in this manner:

1. Assure that the computer PANEL LOCK switch is turned counterclockwise and turn the POWER switch clockwise.

2. Set the Teletype LINE/OFF/LOCAL switch to the LOCAL position.

3. If the punch is to be used, load it by raising the cover, manually feeding the tape from the top of the roll into the guide at the back of the punch, advancing the tape through the punch by manually turning the friction wheel, and then closing the cover. Energize the punch by pressing the ON pushbutton, and produce about two feet of leader. The leader-trailer can be code 200 or 377. To produce the code 200 leader, simultaneously press and hold the CTRL and SHIFT keys with the left hand; press and hold the REPT key; press and release the @ key. When the required amount of leader has been punched release all keys. To produce the 377 code, simultaneously press and hold both the REPT and RUB OUT keys until a sufficient amount of leader has been punched.

If an incorrect key is struck while punching a tape, the tape can be corrected as follows: if the error is noticed after typing and punching N characters, press the punch B. SP. (backspace) pushbutton N + 1 times and strike the keyboard RUB OUT key N + 1 times. Then continue typing and punching with the character which was in error.

To duplicate and obtain a listing of an existing tape: Perform the procedure under the

current heading. Then load the tape to be duplicated as described in step 2 of the procedure under Loading Data Under Program Control. Initiate tape duplication by setting the reader START/STOP/FREE switch in the START position. The punch and teleprinter stop when the tape being duplicated is completely read.

Corrections to insert or delete information on a perforated tape can be made by duplicating the correct portion of the tape, and manually punching additional information or inhibiting punching of information to be deleted. This is accomplished by duplicating the tape and carefully observing the information being typed as the tape is read. In this manner the reader START/STOP/FREE switch can be set to the STOP position just before the point of the correction is typed. Information to be inserted can then be punched manually by means of the keyboard. Information can be deleted by pressing the punch OFF pushbutton and operating the reader until the portion of the tape to be deleted has been typed. It may be necessary to backspace and rub out one or two characters on the new tape if the reader is not stopped precisely on time. The number of characters to be rubbed out can be determined exactly by the typed copy. Be sure to count spaces when counting typed characters. Continue duplicating the tape in the normal manner after making the corrections.

New, duplicated, or corrected perforated tapes should be verified by reading them off line and carefully proofreading the typed copy.

# SECTION D

# INTERFACE AND INSTALLATION

# CHAPTER 1

# PDP-8 INPUT/OUTPUT FACILITIES

Since the processing power of a computer system depends largely upon the range and number of peripheral devices that can be connected to it, the PDP-8 has been designed to interface readily with a broad variety of external equipment. Section D of this handbook defines the interface characteristics of the computer to allow the reader to design and implement any electrical interfaces required to connect devices to the PDP-8. Chapters 2 and 3 functionally describe the logic circuit elements involved in programmed data transfers and data break transfers, respectively. Chapter 4 gives detailed circuit information on the modules used in the computer interface and that are available for use in special device interfaces. Chapter 5 lists connection point, module type, module location, etc., for each interface signal; gives detailed loading and driving characteristics for each module in the computer interface; then presents some general rules and characteristics to be considered in selecting or designing electrical circuits to be connected to the PDP-8. Chapter 6 presents information for planning the installation of a basic PDP-8 and the available standard optional equipment.

The simple I/O techniques of the PDP-8, the availability of DEC's FLIP CHIP logic circuit modules, and DEC's policy of giving assistance wherever possible allow inexpensive, straight-forward device interfaces to be realized. Should questions arise relative to the computer interface characteristics, the design of device interfaces using DEC modules, or installation planning, customers are invited to telephone the main plant in Maynard, Massachusetts, or any of the sales offices. Digital Equipment Corporation makes no representation that the interconnection of its circuit modules in the manner described herein will not infringe on existing or future patent rights. Nor do the descriptions contained herein imply the granting of license to use, manufacture, or sell equipment constructed in accordance therewith.

The basic PDP-8 contains a processor and core memory composed of FLIP CHIP circuit modules. These hybrid silicon circuits have an operating temperature range exceeding the limits of 32° to 130°F, so no air-conditioning is required at the computer site. Standard 115v, 60-cps power operates an internal solid-state power supply that produces all required voltages and currents.

High-capacity, high-speed I/O capabilities of the PDP-8 allow it to operate a variety of peripheral devices in addition to the standard Teletype keyboard/printer, tape reader, and tape punch. DEC options, consisting of an interface and normal data processing equipment, are available for connecting into the computer system. These options include card equipment, line printers, magnetic tape transports, magnetic drums, analog-to-digital converters, CRT displays, and digital plotters. The PDP-8 system can also accept other types of instruments or hardware devices that have an appropriate interface. Up to 61 devices requiring three programmed command pulses, or up to 193 devices requiring one programmed command pulse can be connected to the computer. One machine using the data break facility can be connected directly to the PDP-8, or up to seven such machines can be connected through a Data Multiplexer Type DM01. Interfacing of any devices to the computer requires no modifications to the processor and can be achieved in the field.

Control of some kind is needed to determine when an information exchange is to take place between the PDP-8 and peripheral equipment and to indicate the location(s) in the computer memory which will accept or yield the data. Either the computer program or

the device external to the computer can exercise this control. Transfers controlled by the computer, hence under control of its stored program, are called programmed data transfers. Transfers made at times controlled by the external devices through the data break facility are called data break transfers.

## Programmed Data Transfers

The majority of I/O transfers occur under control of the computer program. To transfer and store information under program control requires about six times as much computer time as under data break control. In terms of real time, the duration of a programmed transfer is rather small, due to the high speed of the computer, and is well beyond that required for laboratory or process control instrumentation.

To realize full benefit of the built-in control features of the PDP-8 programmed I/O transfers should be used in most cases. Controls for devices using programmed data transfers are usually simpler and less expensive than controls for devices using data break transfers. Using programmed data transfer facilities, simultaneous operation of devices is limited only by the relative speed of the computer with respect to the device speeds, and the search time required to determine the device requiring service. Analog-to-digital converters, digital-to-analog converters, digital plotters, line printers, message switching equipment, and realy control systems typify equipment using only programmed data transfers.

## Data Break Transfers

Devices which operate at very high speed or which require very rapid response from the computer use the data break facilities. Use of these facilities permits an external device, almost arbitrarily, to insert or extract words from the computer core memory, bypassing all program control logic. Because the computer program has no cognizance of transfers made in this manner, programmed checks of input data are made prior to use of information received in this manner. The data break is particularly well-suited for devices that transfer large amounts of data in block form, e.g., high-speed magnetic tape systems, high-speed drum memories, or CRT display systems containing memory elements.

## Logic Symbols

Figure 20 defines the symbols used in Section D of this handbook to express signals and digital logic circuits.

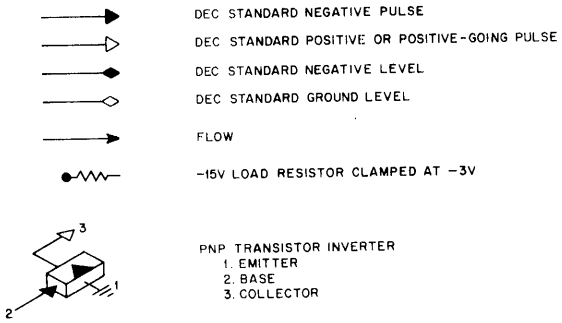



Figure 20 Logic Symbols

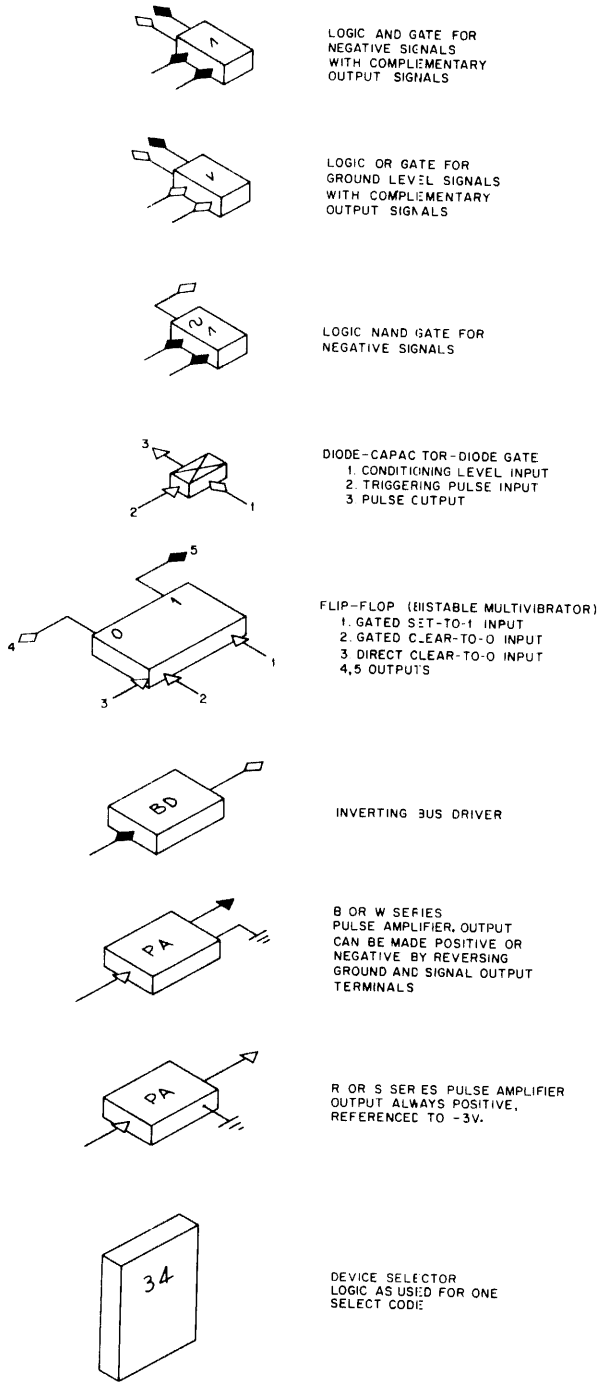



Figure 20 Logic Symbols (continued)

# CHAPTER 2

# PROGRAMMED DATA TRANSFERS

The majority of I/O transfers take place under control of the PDP-8 program, taking advantage of control elements built into the computer. Although programmed transfers take more computer and actual time than do data break transfers, the timing discrepancy is insignificant, considering the high speed of the computer with respect to most peripheral devices. The maximum data transfer rate for programmed operations of 12-bit words is 148 kc when no status checking, end transfer check, etc., is done. This speed is well beyond the normal rate required for typical laboratory or process control instrumentation.

The PDP-8 is a parallel-transfer machine that distributes and collects data in bytes of up to twelve bits. All programmed data transfers take place trhough the accumulator, the 12-bit arithmetic register of the computer. The computer program controls the loading of information into the accumulator (AC) for an output transfer, and for storing information in core memory from the AC for an input transfer. Output information in the AC is power amplified and supplied to the interface connectors for bussed connection to many peripheral devices. Then the program-selected device can sample these signal lines to strobe AC information into a control or information register. Input data arrives at the AC as pulses received at the interface connectors from bussed outputs of many devices. Gating circuits of the program-selected device produce these pulses. Command pulses generated by the device flow to the input/output skip facility (IOS) to sample the condition of I/O device flags. The IOS allows branching of the program based upon the condition or availability of peripheral equipment, effectively making programmed decisions to continue the current program or jump to another part of the program, such as a subroutine that services an I/O device.

The bussed system of input/output data transfers imposes the following requirements on peripheral equipment:

a. The ability of each device to sample the select code generated by the computer during IOT instructions and, when selected, to be capable of producing sequential IOT command pulses in accordance with computer-generated IOP pulses. Circuits which perform these functions in the peripheral device are called the device selector (DS).

b. Each device receiving output data from the computer must contain gating circuits at the input of a receiving register capable of strobing the AC signal information into the register when triggered by a command pulse from the DS.

c. Each device which supplies input data to the computer must contain gating circuits at the output of the transmitting register capable of sampling the information in the output register and supplying a pulse to the computer input bus when triggered by a command pulse from the DS.

d. Each device should contain a busy/done flag (flip-flop) and gating circuits which can pulse the computer input/output skip bus upon command from the DS when the flag is set in the binary 1 state to indicate that the device is ready to transfer another byte of information.

Figure 21 shows the information flow within the computer which effects a programmed data transfer with input/output equipment. All instructions stored in core memory as a program sequence are read into the memory buffer register (MB) for execution. The transfer of the operation code in the three most significant bits (bits 0, 1, and 2) of the instruction into the instruction register (IR) takes place and is decoded to produce appropriate control signals. The computer, upon recognition of the operation code as an IOT instruction, enters a 3.75 μsec expanded computer cycle and enables the IOP generator to produce time sequenced IOP pulses as determined by the three least significant bits of the instruction (bits 9, 10, and 11 in the MB). These IOP pulses and the buffered output of the select code from bits 3-8 of the instruction word in the MB are bussed to device selectors in all peripheral equipment. Figure 22 indicates the timing of programmed data transfers and Figure 23 shows the decoding of the IOT instruction.

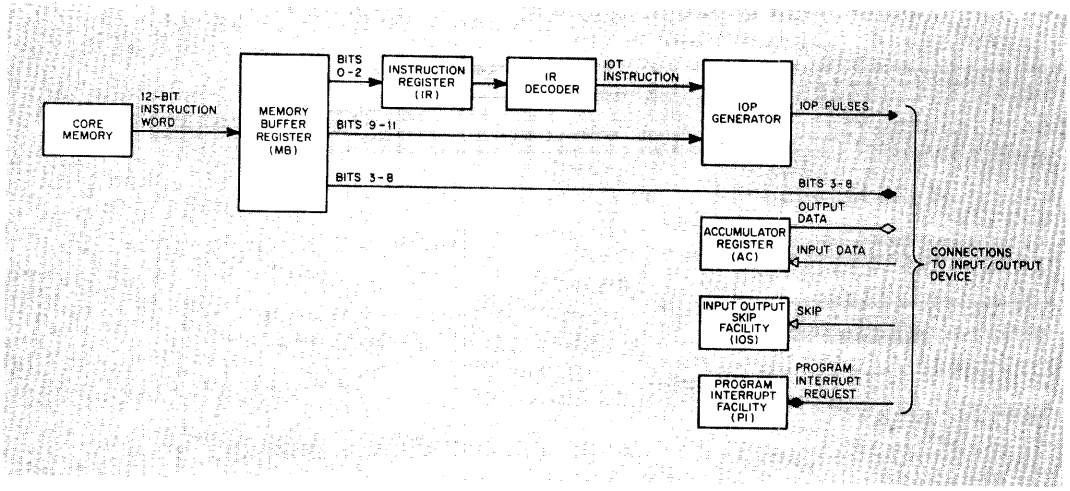



Figure 21 Programmed Data Transfer Interface Block Diagram

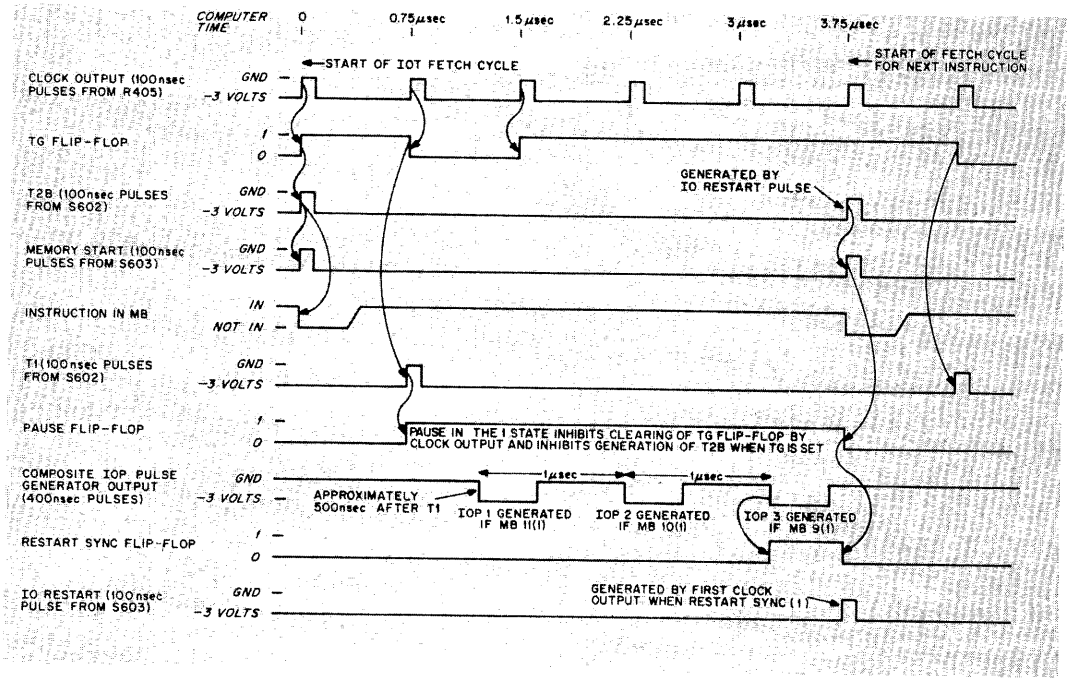



Figure 22 Programmed Data Transfer Timing

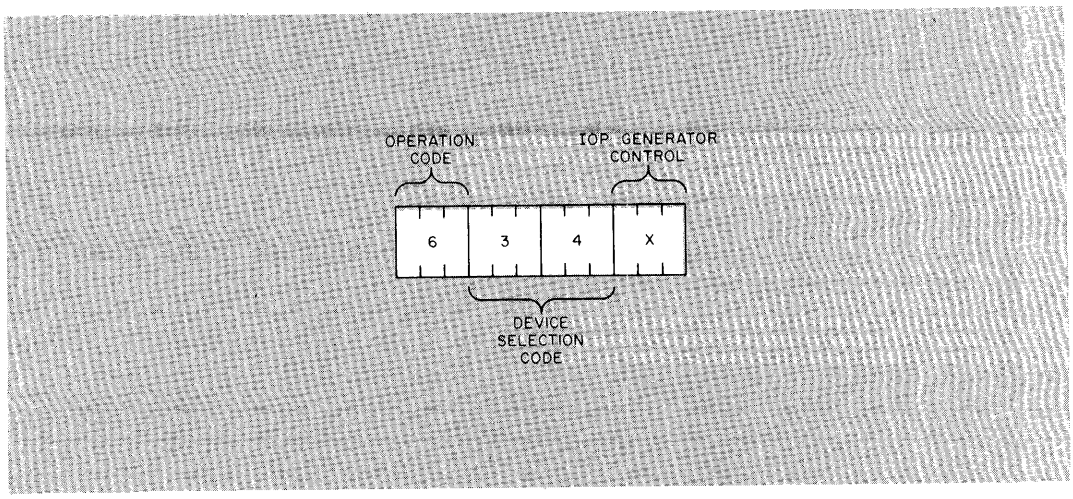



Figure 23 Typical IOT Instruction Decoding

Devices which require immediate service from the computer program, or which take an exorbitant amount of computer time to discontinue the main program until transfer needs are met, can use the program interrupt (PI) facility. In this mode of operation, the computer can initiate operation of I/O equipment and continue the main program until the device requests servicing. A signal input to the PI requesting a program interrupt causes storing of the conditions of the main program and initiates a subroutine to service the device. At the conclusion of this subroutine, the main program is reinstated until another interrupt request occurs.

# Timing and IOP Generator

When the IR decoder detects an operation code of $68$, it identifies an IOT instruction and the computer generates a 1 ⟶ Pause pulse. This pulse disables the normal timing generators of the processor and initiates operation of the IOP generator. The logic circuits of the IOP generator are shown in Figure 24 to consist of three similar channels; each channel consisting of a gated delay, a gated pulse amplifier, and an output pulse amplifier. Operation of the first channel is triggered by the 1 ⟶ Pause pulse and operation of the other two channels is triggered by the pulse output of the delay in the previous channel. Series connection of the delays produces sequential operation of the three channels. The pulse output of the third channel delay restarts the normal timing generators of the processor. Since the time delays are 0.5, 1.0, and 1.0 μsec, the cycle time of an IOT instruction is 3.75 μsec. (IOT instructions associated with enabling and disabling the program interrupt facility, and those for the Analog-to-Digital Converter Type 189, the Memory Extension Control Type 183, and the Data Line Interface Type 681 inhibit generation of the 1 ⟶ Pause pulse and so occur in the normal computer cycle time of 1.5 μsec. In these commands the IOP generator is inhibited so the normal timing pulses of the processor and special device selectors execute these instructions.)

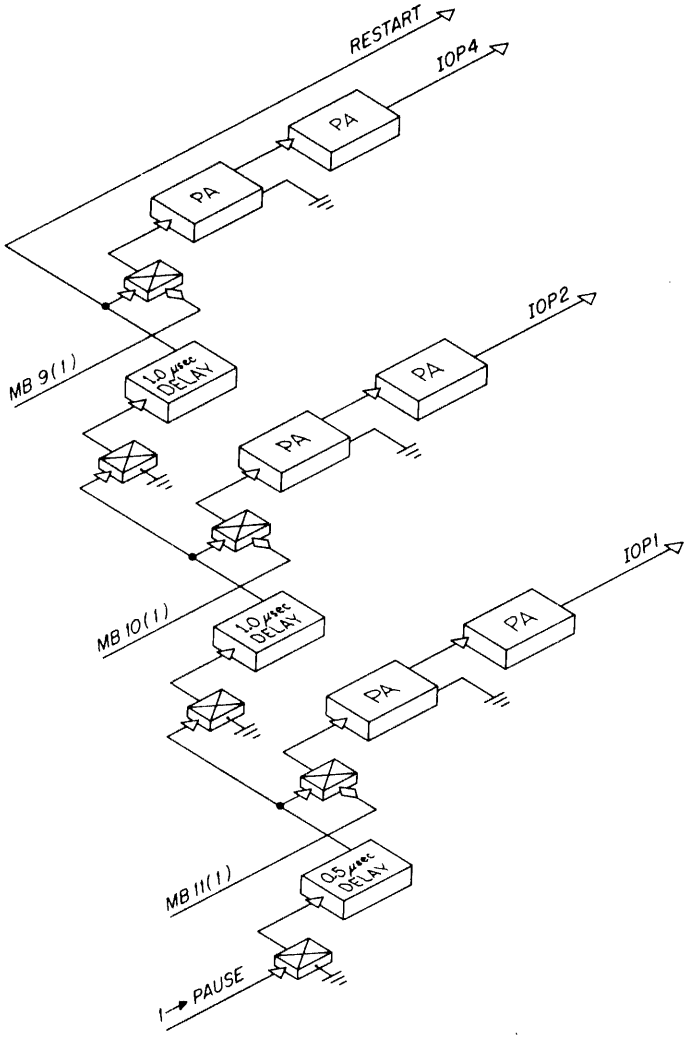



Figure 24 IOP Generator Logic

The gated pulse amplifier of each channel samples the content of one bit of the instruction when the delay output pulse occurs. If the sampled bit contains a binary 1, the gated pulse amplifier is triggered and the output pulse amplifier is operated to produce an IOP pulse. A diode-capacitor-diode (DCD) gate at the input of each gated pulse amplifier serves as a 2-input AND gate. The binary 1 status of one of the least significant bits of the instruction in the MB supplies the conditioning level of each of these gates. The output of the gated pulse amplifier initiates operation of the output pulse amplifier to generate an IOP pulse which is available at the interface connector as a DEC standard 0.4 μsec negative pulse. This configuration allows each IOP pulse to be individually programmed, permits a sequence of up to three events to occur within each instruction, and provides 1 μsec between events for normal device circuit set-up times. The instruction bit that enables or disables generation of each IOP pulse, the corresponging number of the IOT pulse produced in the DS from the IOP pulse, and the event time for each IOP pulse is:

| Instruction Bit | IOP Pulse | IOT Pulse | Event Time |
|---|---|---|---|
| 11 | IOP 1 | IOT 1 | 1 |
| 10 | IOP 2 | IOT 2 | 2 |
| 9 | IOP 4 | IOT 4 | 3 |

# Device Selector (DS)

Bits 3 through 8 of an IOT instruction serve as a device or subdevice select code. Bus drivers in the processor buffer both the binary 1 and 0 output signals of MB3-8 and distribute them to the interface connectors for bussed connection to all device selectors. Each DS is assigned a select code and is enabled only when the assigned code is present in the MB. When enabled, a DS regenerates IOP pulses as IOT command pulses and transmits these pulses to skip, input, or output gates within the device and/or to the processor to clear the AC.

Each group of three command pulses requires a separate DS channel (W103 module), and each DS channel requires a different select code (or I/O device address). One I/O device can, therefore, use several DS channels. Note that the processor produces the pulses identified as IOP 1, IOP 2, and IOP 4 and supplies them to all device selectors. The device selector produces pulses IOT 1, IOT 2, and IOT 4 which initiate a transfer or effect some control. Figure 25 shows generation of command pulses by several DC channels.

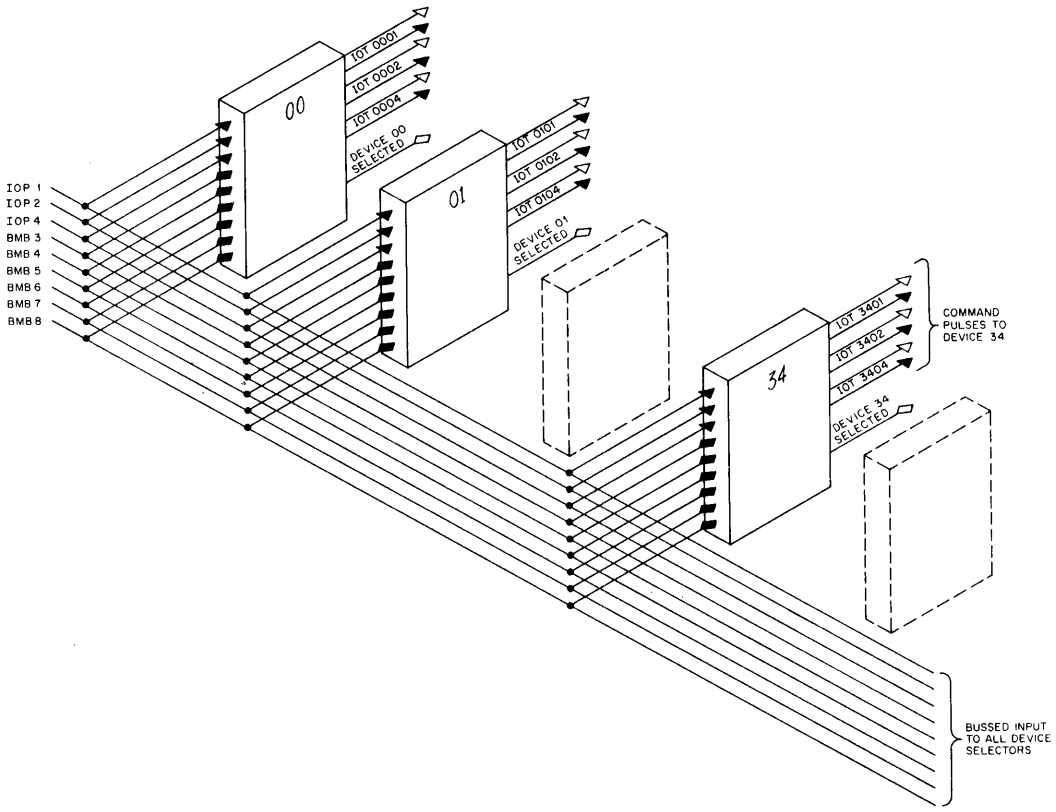



Figure 25 Generation of IOT Command Pulses by Device Selectors

The logical representation for a typical channel of the DS, using channel 34, is shown in Figure 26. A 6-input NAND gate wired to receive the appropriate signal outputs from MB3-8 for select code 34 activates the channel. In the DS module, the NAND gate contains 14 diode input terminals; 12 of these connect to the complementary outputs of MB3-8, and 2 are open to receive subdevice or control condition signals as needed. Either the 1 or the 0 signal from each MB bit is disconnected by removing the appropriate diode from the NAND gate when establishing the select code. The ground level output of the NAND gate indicates when the IOT instruction selects the device, and can therefore enable circuit operations within the device. This output also enables three gating inverters, allowing them to trigger a pulse amplifier if an IOP pulse occurs. The positive output from each pulse amplifier is an IOT command pulse identified by the select code and the number of the initiating IOP pulse. Three inverters receive the positive IOT pulses to produce complementary IOT output pulses. A pulse amplifier module can be connected in each channel of the DS to provide greater output drive or to produce pulses of a specific duration required by the selected device.

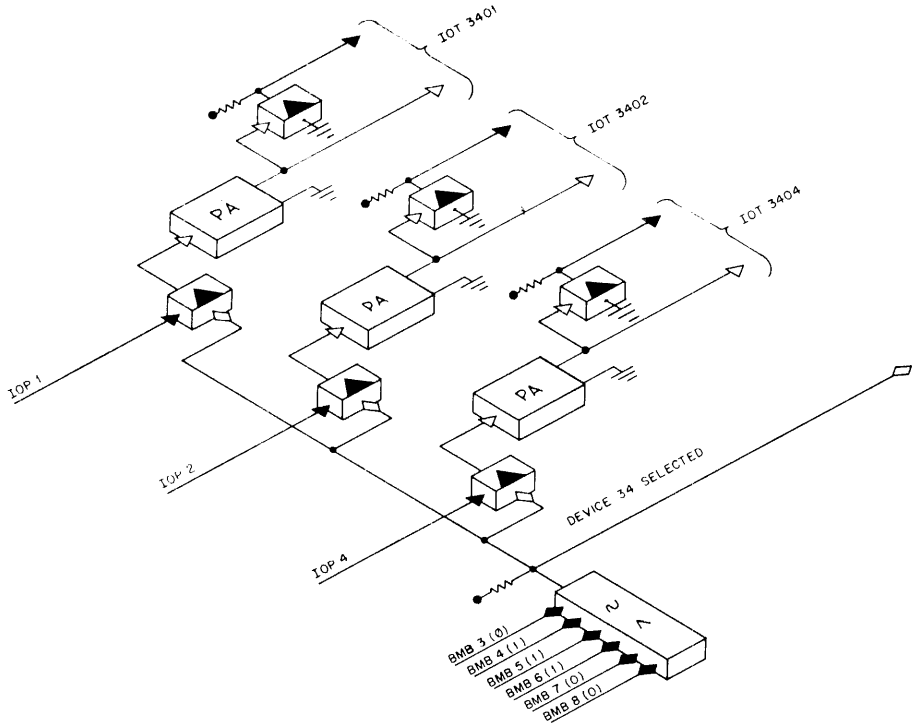



Figure 2o Typical Device Selector (Device 34)

# Input/Output Skip (IOS)

Generation of an IOT pulse can be used to test the condition or status of a device flag, and to continue to or skip the next sequential instruction based upon the results of this test. This operation is performed by a 2-input AND gate in the device connected as shown in Figure 27. One input of the skip gate receives the status level (flag output signal), the second input receives an IOT pulse, and the output drives the computer IOS bus to ground when the skip conditions are fulfilled. When the IOS bus is driven to ground, the content of the program counter is incremented by 1 to advance the program

count without executing the instruction at the current program count. In this manner an IOT instruction can check the status of an I/O device flag and skip the next instruction if the device requires servicing. Programmed testing in this manner allows the routine to jump out of sequence to a subroutine that services the device tested.

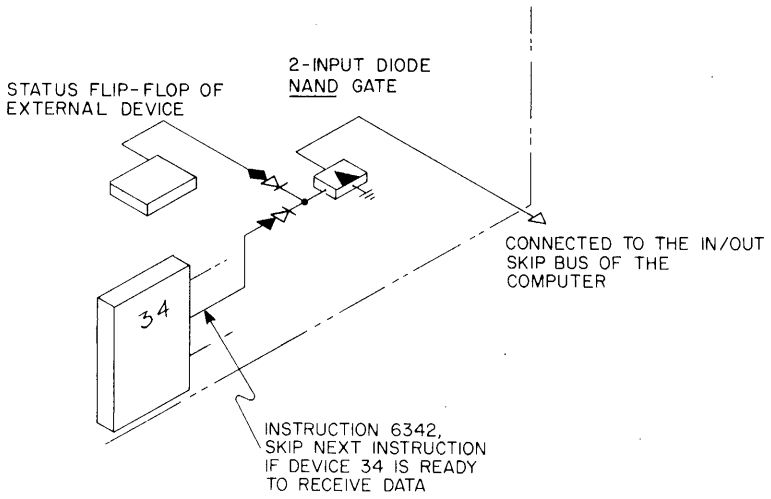



Figure 27 Use of IOS to Test the Status of an External Device

Assuming that a device is already operating, a possible program sequence to test its availability follows:

| Address | Instruction | Remarks |
|---|---|---|
| . | | |
| . | | |
| . | | |
| 100, | 6342 | /SKIP IF DEVICE 34 IS READY |
| 101, | 5100 | /JUMP .-1 |
| 102, | 5XXX | /ENTER SERVICE ROUTINE FOR |
| | | /DEVICE 34 |
| . | | |
| . | | |
| . | | |

When the program reaches address 100, it executes an instruction skip with 6342. The skip occurs only if device 34 is ready when the IOT 6342 command is given. If device 34 is not ready, the flag signal disqualifies the skip gate, and the Skip pulse does not occur. Therefore, the program continues to the next instruction which is a jump back to the skip instruction. In this example, the program stays in this waiting loop until the device is ready to transfer data, at which time the skip gate in the device is enabled and the Skip pulse is sent to the computer IOS facility. When the skip occurs, the instruction in location 102 transfers program control to a subroutine to service device 34. This subroutine can load the AC with data and transfer it to device 34, or can load the AC from a register in device 34 and store it in some known core memory address.

## Accumulator

The binary 1 output signal of each flip-flop of the AC, buffered by a bus driver, is available at the interface connectors. These computer data output lines are bus connected to

all peripheral equipment receiving programmed data output information from the PDP-8. A direct-set terminal on each flip-flop of the AC is connected to the interface connectors for bussing to all peripheral equipment supplying programmed data input to the PDP-8. A pulse that drives the direct-set terminal to ground causes setting of an AC flip-flop to the binary 1 state. Output and input connections to the accumulator appear in Figure 28.

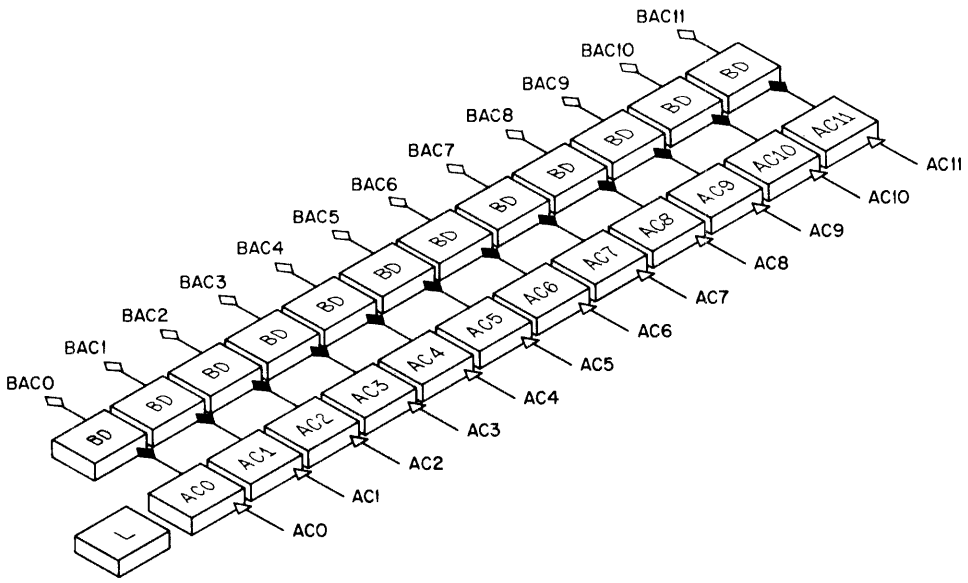



Figure 28 Accumulator Input and Output

Figure 28 illustrates the twelve bits of the accumulator and the link bit. The status of the link bit is not available to enter into transfers with peripheral equipment (unless it is rotated into the AC). A noninverting bus driver continuously buffers the output signal from each AC flip-flop. These buffered accumulator (BAC) signals are available at the interface connectors.

# Input Data Transfers

When ready to transfer data into the PDP-8 accumulator, the device sets a flag connected to the IOS. The program senses the ready status of the flag and issues an IOT instruction to read the content of the external device buffer register into the AC. If the AC is not cleared before the transfer, the resultant word in the AC is the inclusive OR of the previous word in the AC and the word transferred from the device buffer register.

The illustration in Figure 29 shows that the accumulator has an input bus for each bit flip-flop. Setting a 1 into a particular bit of the accumulator necessitates grounding of the interface input bus by the standard DEC inverter. In the illustration, the 2-input AND gates set various bits of the accumulator. In this case an IOT pulse is AND combined with the flip-flop state of the external device to conditionally set 1's into the accumulator. (The program must include a clear AC command prior to loading in this manner; otherwise an inclusive OR takes place between the previous content of the accumulator and the content of the data register being read.)

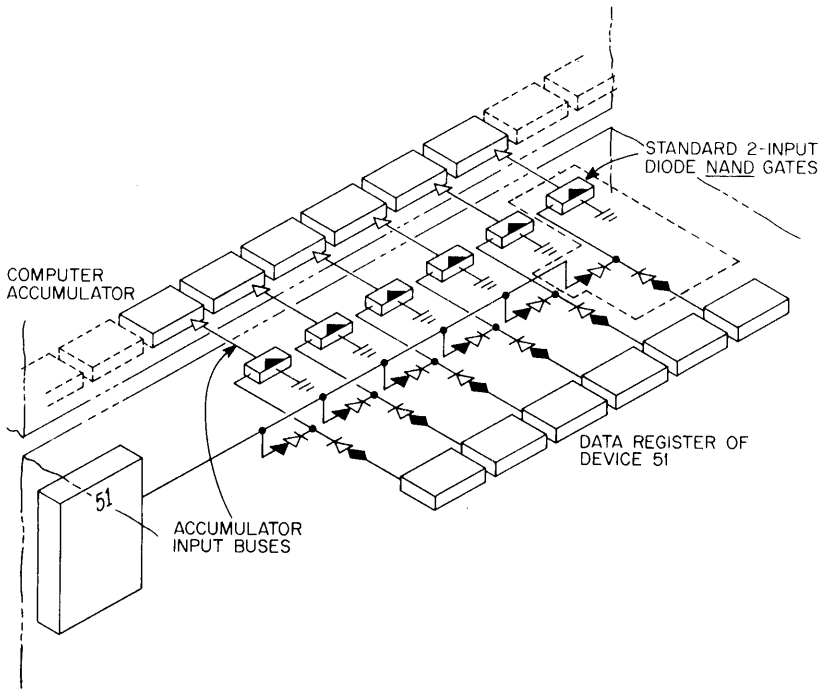



Figure 29 Loading Data into the Accumulator from an External Device

Following the transfer (possibly in the same instruction) the program can issue a command pulse to initiate further operation of the device and/or clear the device flag.

# Output Data Transfers

The AC is loaded with a word (e.g., by a CLA TAD instruction sequence); then the IOT instruction is issued to transfer the word into the control or data register of the device by an IOT pulse (e.g., IOP 2), and operation of the device is initiated by another IOT pulse (e.g., IOP 4). The data word transferred in this manner can be a character to be operated upon, or can be a control word sampled by a status register to establish a control mode.

Since the BAC interface bus lines continually represent the status of the AC flip-flops, the receiving device can strobe them to sense the value in the accumulator. In Figure 30 a strobe pulse samples six bits of the accumulator to conditionally set an external 6-bit data register. Since this is not a jam transfer, it is necessary first to clear the external data register before setting 1's into it. The readin gates driving the external data register are part of the external device and are not supplied by the computer. The data register can contain any number of flip-flops up to a maximum of twelve. (If more than twelve flip-flops are involved, two or more transfers must take place.) Obviously the clear pulse and the strobe pulse shown in Figure 30 must occur when the data to be placed in the external data register is held in the accumulator. These pulses therefore must be under computer control to effect synchronization with the operation or program of the computer.

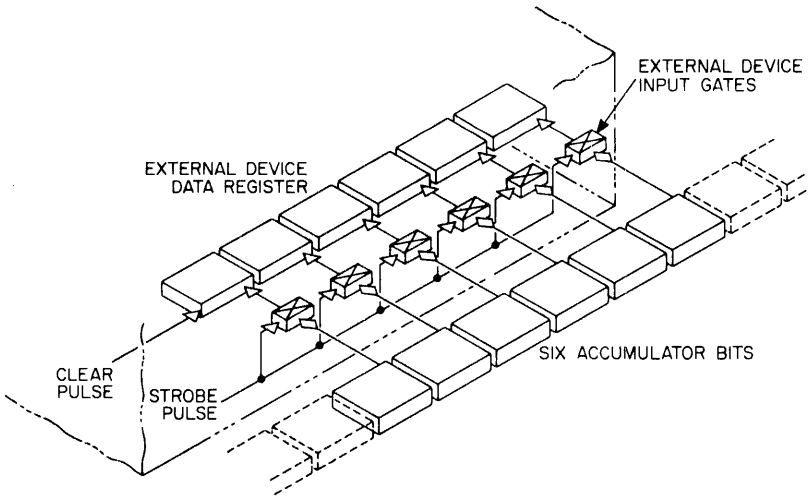



Figure 30 Loading a 6-Bit Word into an External Device from the Accumulator

Figure 31 illustrates the use of two of the pulses being gated by the device selector coded for "34." Pulse IOT 1 clears the data register and IOT 4 strobes the data from the accumulator into the data register. Note that the processor produces the IOP 1, IOP 2, and IOP 4 pulses and supplies them to all device selectors. The program-selected DS produces IOT 1, IOT 2, and IOT 4 pulses which initiate a transfer or effect some control. As indicated in Figure 31 this particular system adds two new microinstructions to the PDP-8 repertoire. One generates a pulse to clear the data register of device number 34. The other microinstruction produces a pulse to load the data register of device number 34 with the content of the accumulator.

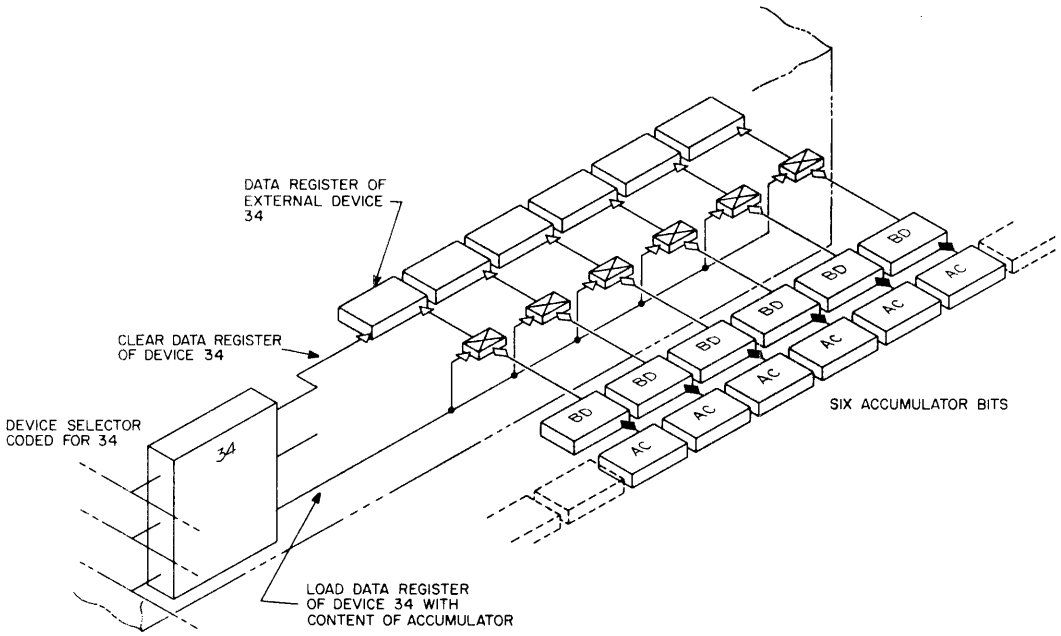



Figure 31 Use of a Device Selector for Activating and Controlling an External Device

The timing of the IOT cycle is shown in Figure 22. Note that the AC bus drivers are quiescent 400 nsec before the IOP 1 pulse occurs. Since FLIP CHIP DCD gates require a 400-nsec set-up time, the IOP 1 pulse cannot be used to load the content of the AC into an external buffer register having input DCD gates. If the device register has DCD gates, IOP 1 should be used to reset or clear registers, controls, or flags. The IOP 1 pulse can be used to read the content of the AC into an external device register that is equipped with input diode gates. IOP 2 or IOP 4 can be used to strobe the content of the AC through DCD gates if the lead lengths of the BAC lines and the pulse lines provide equivalent transmission delays. Only IOP 1 or IOP 2 (not IOP 4) can be used with the IOS facility.

# Program Interrupt (PI)

When a large amount of computing is required, the program should initiate operation of an I/O device then continue the main program, rather than wait for the device to become ready to transfer data. The program interrupt facility, when enabled by the program, relieves the main program of the need for repeated flag checks by allowing the ready status of I/O device flags to automatically cause a program interrupt. When the program interrupt occurs, program control transfers to a subroutine that determines which device requested the interrupt and initiates an appropriate service routine.

In the example shown in Figure 32, a flag signal from a status flip-flop operates a standard inverter with no collector load. When the status flip-flop indicates the need for device service, the inverter drives the Program Interrupt Request bus to ground to request a program interrupt.

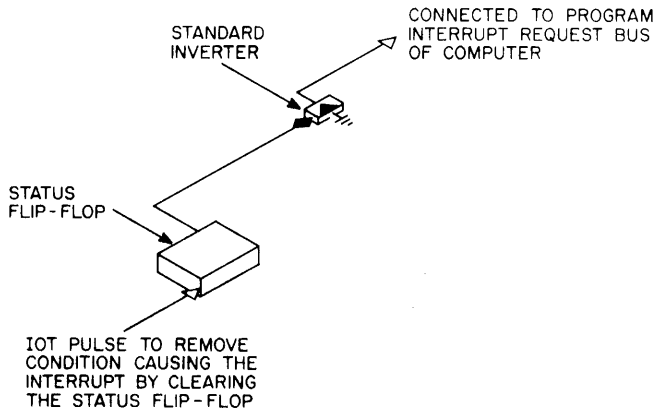



Figure 32 Program Interrupt Request Signal Origin

If only one device is connected to the PI facility, program control can be transferred directly to a routine that services the device when an interrupt occurs. This operation occurs as follows:

| Tag | Address | Instruction | Remarks |
|---|---|---|---|
| | 1000 | . | /MAIN PROGRAM |
| | 1001 | . | /MAIN PROGRAM CONTINUES |
| | 1002 | . | /INTERRUPT REQUEST OCCURS |
| | | INTERRUPT OCCURS | |
| | 0000 | | /PROGRAM COUNT (PC=1003) IS STORED IN 0000 |
| | 0001 | JMP SR | /ENTER SERVICE ROUTINE |
| SR | 2000 | . | /SERVICE SUBROUTINE FOR INTERRUPTING |
| | | . | /DEVICE AND SEQUENCE TO RESTORE |
| | 3001 | . | /AC, AND RESTORE L IF REQUIRED |
| | 3002 | ION | /TURN ON INTERRUPT |
| | 3003 | JMP I 0000 | /RETURN TO MAIN PROGRAM |
| | 1003 | . | /MAIN PROGRAM CONTINUES |
| | 1004 | . | |
| | | . | |
| | | . | |

In most PDP-8 systems numerous devices are connected to the PI facility, so the routine beginning in core memory address 0001 must determine which device requested an interrupt. The interrupt routine determines the device requiring service by checking the flags of all equipment connected to the PI and transfers program control to a service routine for the first device encountered that has its flag in the state required to request a program interrupt. In other words, when program interrupt requests can originate in numerous devices, each device flag connected to the PI must also be connected to the IOS.

# Multiple Use of IOS and PI

In common practice, more than one device is connected to the PI facility. Therefore, since the computer receives a request that is the inclusive OR of requests from all devices connected to the PI, the IOS must identify the device making the request. When a program interrupt occurs, a routine is entered from address 0001 to sequentially check the status of each flag connected to the PI and to transfer program control to an appropriate service routine for the device whose flag is requesting a program interrupt. Figure 33 shows IOS and PI connections for three typical devices.

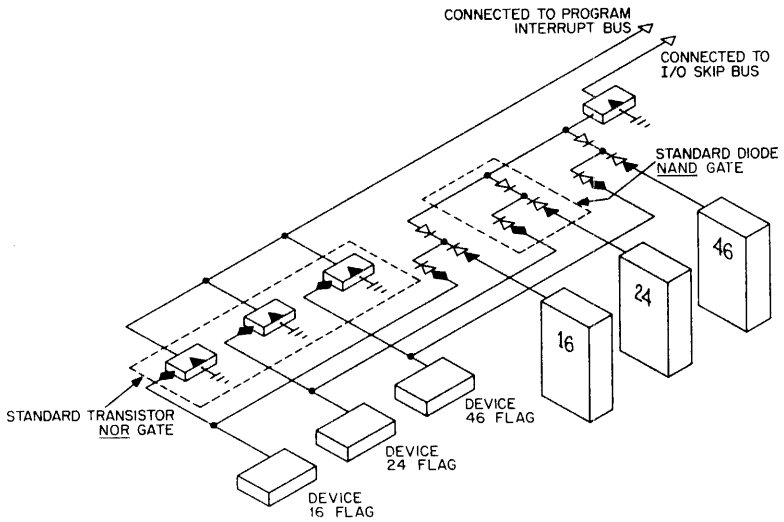



Figure 33 Multiple Inputs to IOS and PI Facilities

The following program example illustrates how the program interrupt routine determines the device requesting service:

```
Tag       Address   Instruction        Remarks

          1000        .          /MAIN PROGRAM
          1001        .          /MAIN PROGRAM CONTINUES
          1002        .          /INTERRUPT REQUEST OCCURS
                        INTERRUPT OCCURS
          0000                   /STORE PC (PC = 1003)
          0001      JMP FLG CK   /ENTER ROUTINE TO DETERMINE WHICH DEVICE
                                 /CAUSED INTERRUPT
FLG CK              IOT 6341     /SKIP IF DEVICE 34 IS REQUESTING
                    SKP          /NO - TEST NEXT DEVICE
                    JMP SR34     /ENTER SERVICE ROUTINE 34
                    IOT 6441     /SKIP IF DEVICE 44 IS REQUESTING
                    SKP          /NO - TEST NEXT DEVICE
                    JMP SR44     /ENTER SERVICE ROUTINE 44
                    IOT 6541     /SKIP IF DEVICE 54 IS REQUESTING
                    SKP          /NO - TEST NEXT DEVICE
                    JMP SR54     /ENTER SERVICE ROUTINE 54
                      .
                      .
                      .
```

Assume that the device that caused the interrupt is an input device (e.g., tape reader). The following example of a device service routine might apply:

```
Tag               Instruction        Remarks

SR                DAC TEMP     /SAVE AC
                  IOT XX       /TRANSFER DATA FROM DEVICE BUFFER TO AC
                  DAC I 10     /STORE IN MEMORY LIST
                  ISZ COUNT    /CHECK FOR END
                  SKP          /NOT END
                  JMP END      /END. JUMP TO ROUTINE TO HANDLE END OF
                                /LIST CONDITION
                    .
                    .
                    .
                               /RESTORE L AND EPC IF REQUIRED
                  TAD TEMP     /RELOAD AC
                  ION          /TURN ON INTERRUPT
                  JMP I 0      /RETURN TO PROGRAM
```

If the device that caused the interrupt was essentially an output device (receiving data from computer), the IOT - then - DAC I 10 sequence might be replaced by a TAD I 10 - then - IOT sequence.

# CHAPTER 3

# DATA BREAK TRANSFERS

The data break facility allows one I/O device to transfer information directly with the PDP-8 core memory on a cycle-stealing basis. Up to seven devices can connect to the data break facility through the optional Data Multiplexer Type DM01. The data break is particularly well-suited for devices which transfer large amounts of information in block form.

Peripheral I/O equipment operating at high speeds can transfer information with the computer through the data break facility more efficiently than through programmed means. The combined maximum transfer rate of the data break facility is over 7.8 million bits per second. Information flow to effect a data break transfer with an I/O device appears in Figure 34.

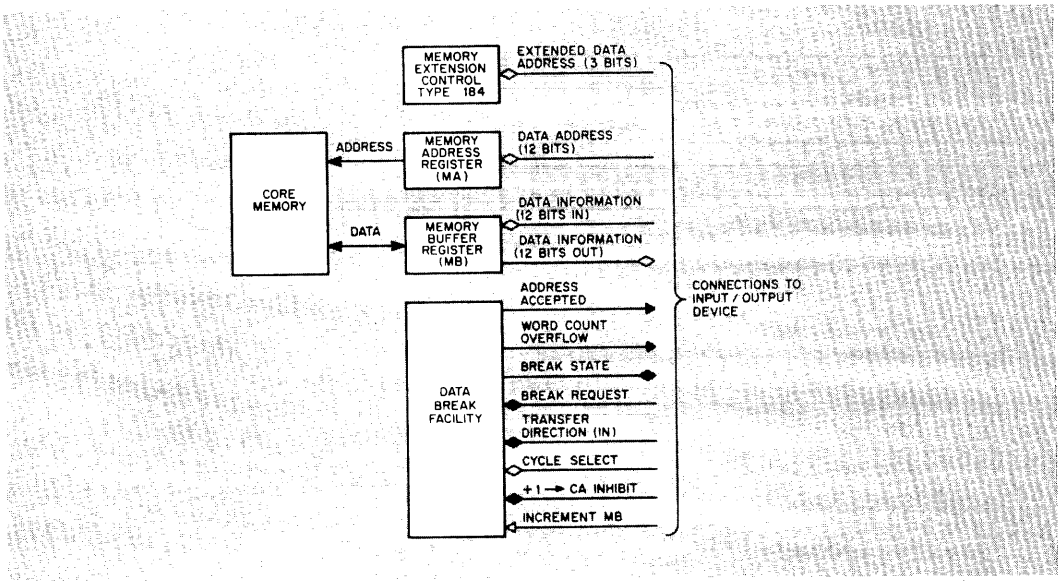



Figure 34 Data Break Transfer Interface Block Diagram

In contrast to programmed operations, the data break facilities permit an external device to control information transfers. Therefore, data-break device interfaces require more control logic circuits, causing a higher cost than programmed-transfer interfaces.

Data breaks are of two basic types: single-cycle and three-cycle. In a single-cycle data break, registers in the device (or device interface) specify the core memory address of each transfer and count the number of transfers to determine the end of data blocks. In the three-cycle data break two computer core memory locations perform these functions, simplifying the device interface by omitting two hardware registers.

In general terms, to initiate a data break transfer of information, the interface control must do the following:

a. Specify the affected address in core memory.

b. Provide the data word by establishing the proper logic levels at the computer interface (assuming an input data transfer), or provide readin gates and storage for the word (assuming an output data transfer).

c. Provide a logical signal to indicate direction of data word transfer.

d. Provide a logical signal to indicate single-cycle or three-cycle break operation.

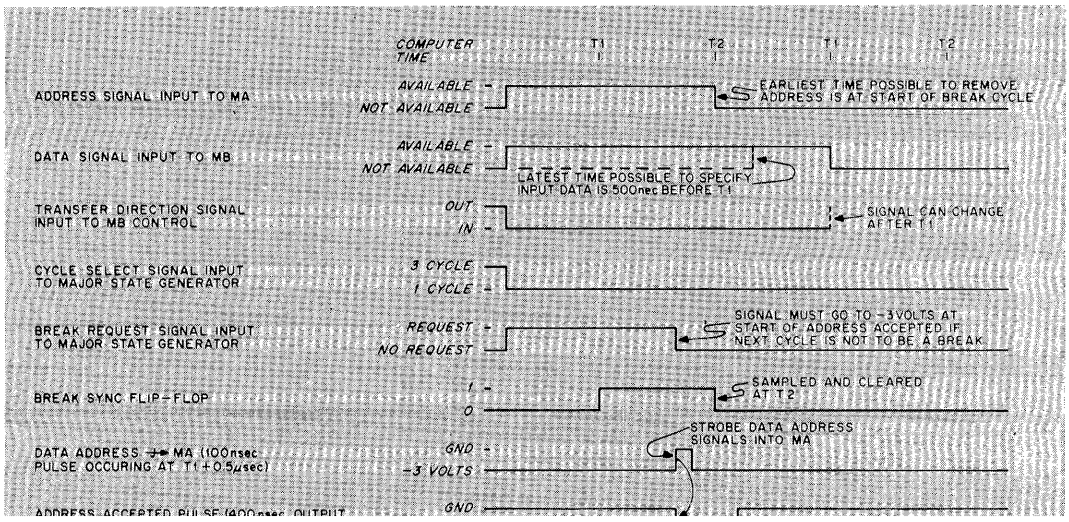
e. Request a data break by supplying a proper signal to the computer data break facility.

# Single-Cycle Data Breaks

Single-cycle breaks are used for input data transfers to the computer, output data transfers from the computer, and memory increment data breaks. Memory increment is a special output data break in which the content of a memory address is read, incremented by 1, and rewritten at the same address. It is useful for counting iterations or external events without disturbing the computer program counter (PC) or AC registers.

## INPUT DATA TRANSFERS

Figure 35 illustrates timing of an input transfer data break. The address to be affected in core is normally provided in the device interface in the form of a 12-bit flip-flop register (data break address register) which has been preset by the interface control by programmed transfer from the computer.



External registers and control flip-flops supplying information and control signals to the data break facility and other PDP-8 interface elements are shown in Figure 36. The input buffer register (IB in Figure 36) holds the 12-bit data word to be written into the computer core memory location specified by the address contained in the address register (AR in Figure 36). Appropriate output terminals of these registers are connected to the computer to supply ground potential to designate binary 1's. Since most devices that transfer data through the data break facility are designed to use either single-cycle or three-cycle breaks, but not both, the Cycle Select signal can usually be supplied from a stable source (such as a ground connection or a −3v clamped load resistor) rather than from a bistable device as shown in Figure 36.

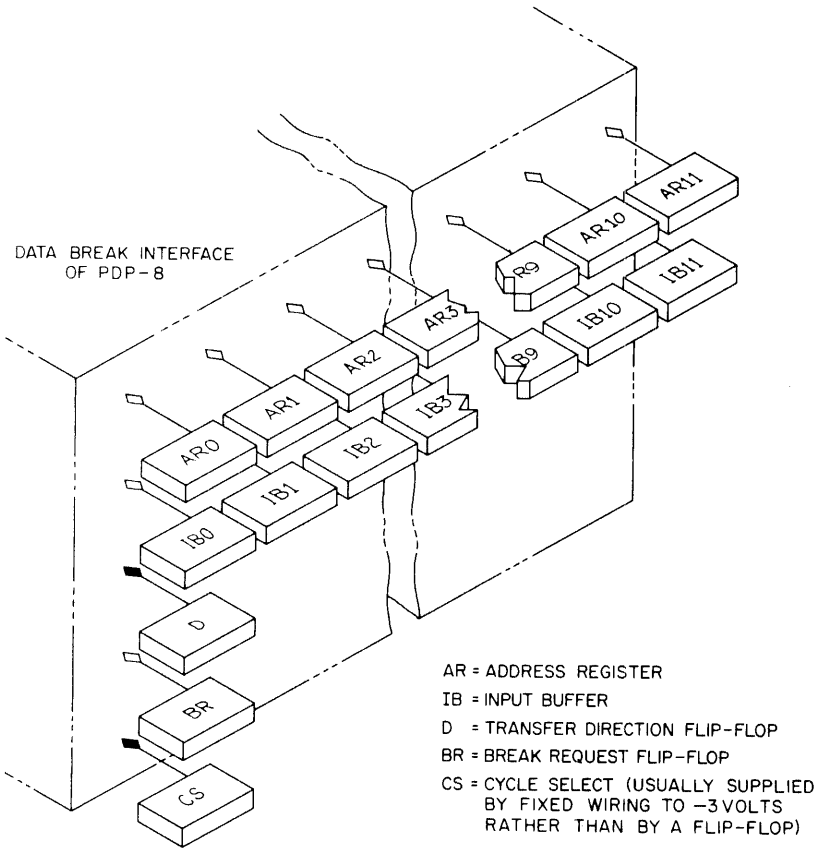


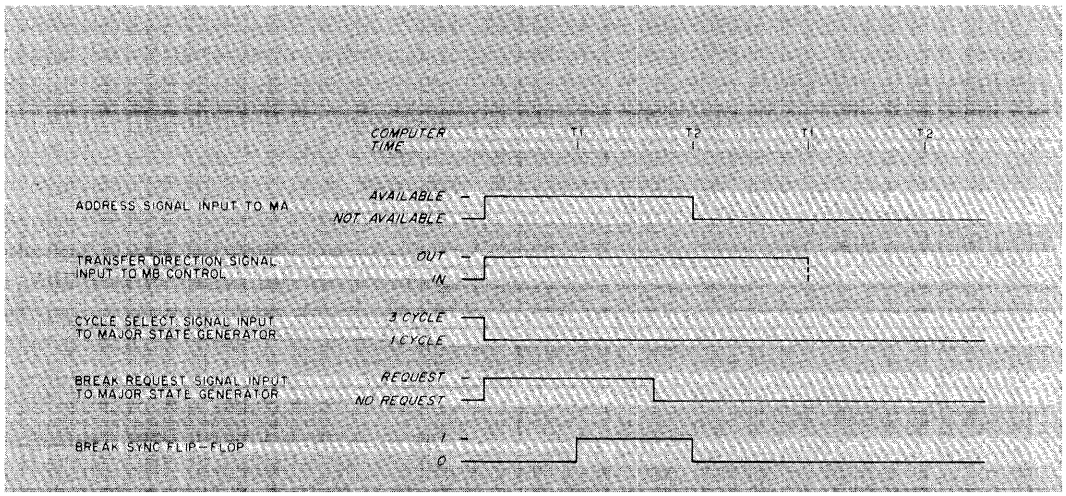

Figure 36 Device Interface Logic for Single-Cycle Data Break Input Transfer

Other portions of the device interface, not shown in Figure 36, establish the data word in the input buffer register, set the address into the address register, set the direction flip-flop to indicate an input data transfer, and control the break request flip-flop. These operations can be performed simultaneously or sequentially, but all transients should occur before the data break request is made. Note that the device interface need supply only static levels to the computer, minimizing the synchronizing logic circuits necessary in the device interface.

When the data break request arrives, the computer completes the current instruction, generates an Address Accepted pulse (at T1 time of the cycle preceding the data break) to acknowledge receipt of the request, then enters the Break state to effect the transfer (see Section A, Chapter 5 of this handbook for more details on data break operations performed by the computer). The Address Accepted pulse can be used in the device interface to clear the break request flip-flop, increment the content of the address register, etc. If the Break Request signal is removed before T1 time of the data break cycle, the computer performs the transfer in one 1.5-μsec cycle and returns to programmed operation.

## OUTPUT DATA TRANSFERS

Timing of operations occurring in a single-cycle output data break is shown in Figure 37. Basic logic circuits for the device interface used in this type of transfer are shown in Figure 38. Address and control signal generators are similar to those discussed previously for input data transfers, except that the Transfer Direction signal must be at ground potential to specify the output transfer of computer information. An output data register (OB in Figure 38) is usually required in the device interface to receive the computer information. The device, and not the PDP-8, controls strobing of data into this register. The device must supply strobe pulses for all data transfers out of the computer (programmed or data break) since circuit configuration and timing characteristics differ in each device.



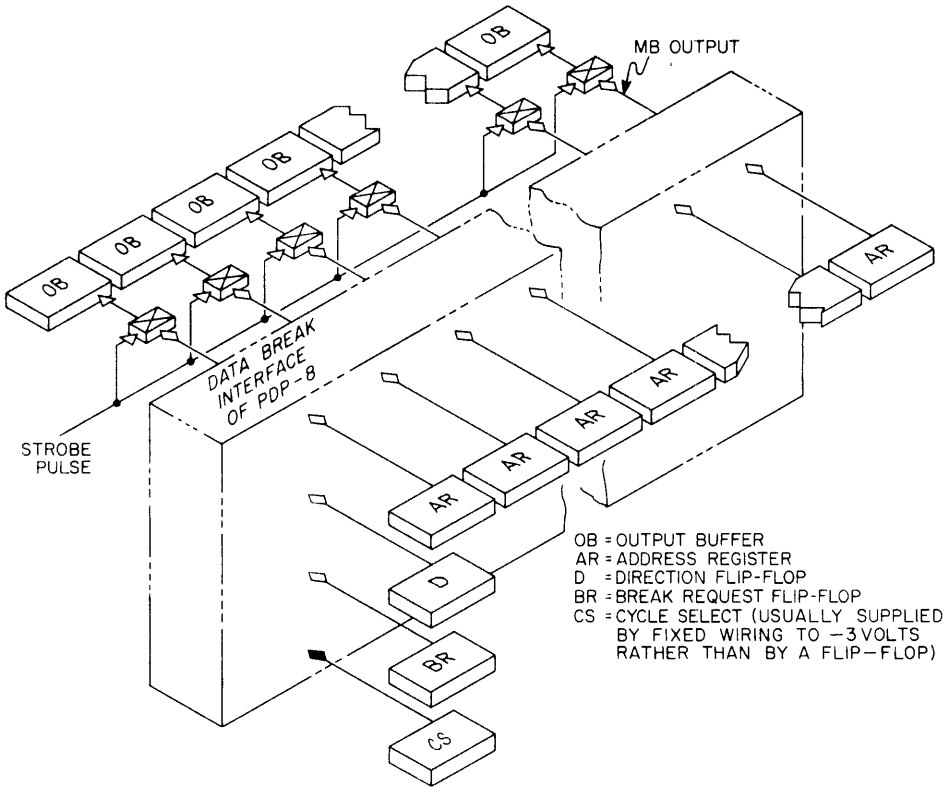



Figure 38 Device Interface Logic
for Single-Cycle Data Break Output Transfer

When the data break request arrives, the computer completes the current instruction and generates an Address Accepted pulse as in input data break transfers. At T2 time the address supplied to the PDP-8 is loaded into the MA, the Break state is entered, and the MB is cleared. Not more than 350 nsec after T2, the content of the device-specified core memory address is read and available in the MB. (This word is automatically rewritten at the same address during the last half of the Break cycle and is available for programmed operations when the data break is finished.) Data Bit signals are available as static levels of ground potential for binary 1's and −3v for binary 0's. The MB is cleared at T2 time of each computer cycle, so the data word is available in the MB for approximately 1.15 μsec to be strobed by the device interface.

Generation of the strobe pulse by the device interface can be synchronized with computer timing through use of timing pulses B T1 or B T2A, which are available at the computer interface. In addition to a timing pulse (delayed or used directly from the computer), generation of this strobe pulse should be gated by condition signals that occur only during the Break cycle of an output transfer. Figure 39 shows typical logic circuits to effect an output data transfer. In this example the B Break signal and an inverted Transfer Direction signal are combined in a diode NAND gate to condition a diode-capacitor-diode gate. A buffered B T2A pulse triggers the DCD gate to produce the strobe pulse. The B T2A pulse determines the timing of the transfer in this example, since the input of the output buffer register has DCD gates. Conventional DCD gates require a minimum set-up time of 400 nsec, which is adequately provided between the time when data is avail-

able in the MB and T2 time. Although the MB is cleared and the major state generator is changed at T2 time, the B T2A pulse can effect this transfer because the delay built into FLIP CHIP flip-flops allows the output to be sampled while the input is being pulsed. If diode gates or other devices with a set-up time of less than 400 nsec are used at the input of the output buffer register, the B T1 pulse, or some other pulse generated by the device interface before T2 time, can trigger strobe pulse generation.

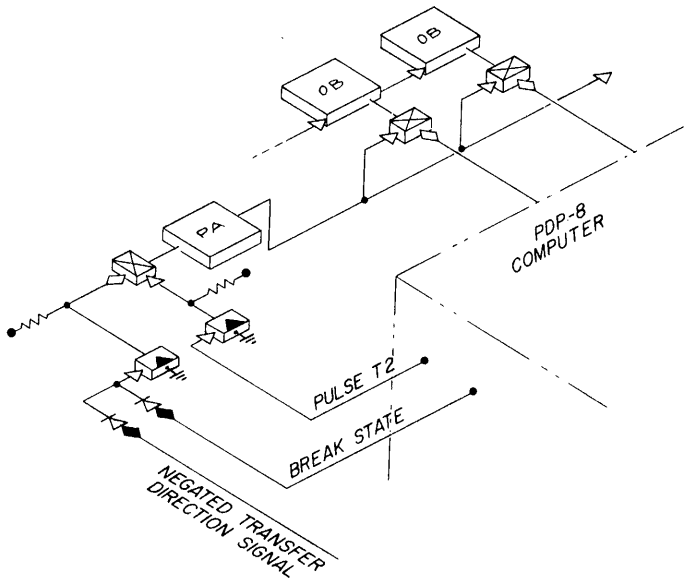



Figure 39 Device Interface for Strobing Output Data

By careful design of the input and output gating, one register can serve as both the input and output buffer register. Most DEC options using the data break facility have only one data buffer register with appropriate gating to allow it to serve as an output buffer when the Transfer Direction signal is at ground potential or as an input buffer when the Transfer Direction signal is −3v.

## MEMORY INCREMENT

In this type of data break the content of core memory at a device-specified address is read into the MB, is incremented by 1, and is rewritten at the same address within one 1.5-μsec cycle. This feature is particularly useful in building a histogram of a series of measurements, such as in pulse-height analysis applications. For example, in a computer-controlled experiment that counts the number of times each value of a parameter is measured, a data break can be requested for each measurement, and the measured value can be used as the core memory address to be incremented (counted).

Signal interface for a memory increment data break is similar to an output transfer data break except that the device interface generates an Increment MB signal and does not generate a strobe pulse (no data transfer occurs between the PDP-8 and the device). Timing of memory increment operations appear in Figure 40, and an example of the logic circuits used by a device interface appears in Figure 41.

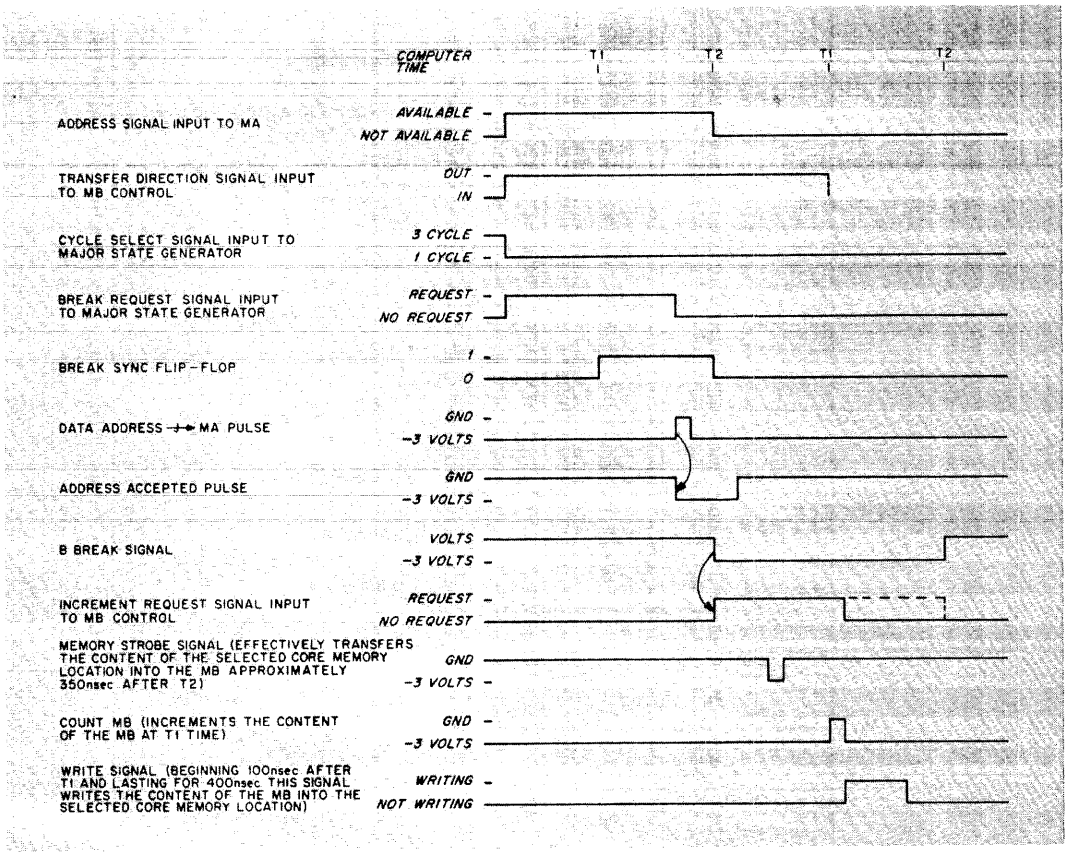
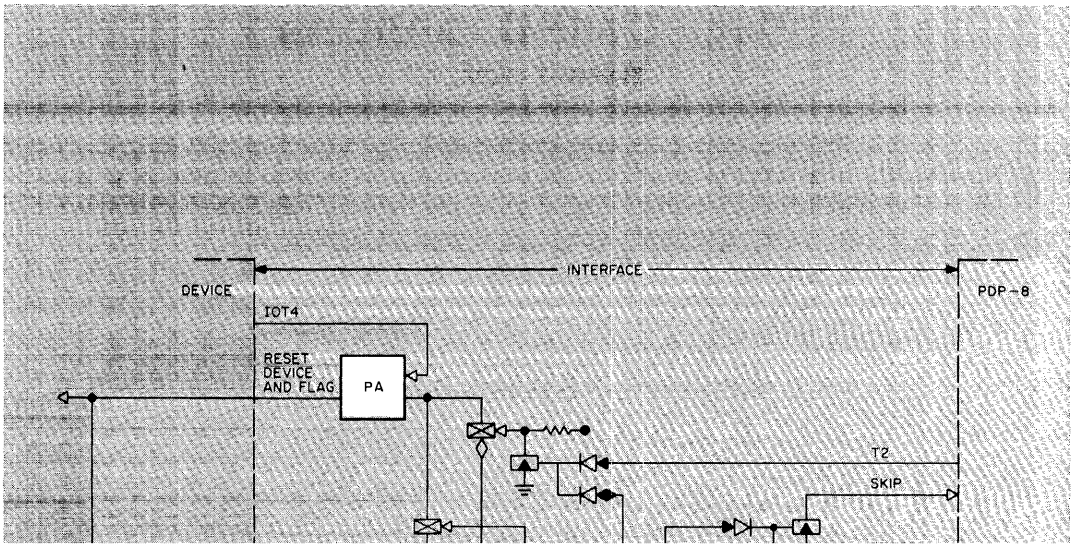



Figure 40 Memory Increment Data Break Timing Diagram

An interface for a device using memory increment data breaks must supply twelve Data Address signals, a Transfer Direction signal, a Cycle Select signal, and a Break Request signal to the computer data break facility as in an output transfer data break. In addition, a ground potential Increment MB signal must be provided at least 400 nsec before T1 time of the Break cycle. This signal can be generated in the device interface by AND combining the B Break computer output signal, the output transfer condition of the Transfer Direction signal, and the condition signal in the device that indicates that an increment operation should take place. When the computer receives this Increment MB signal, it forces the MB control element to generate a Count MB pulse at T1 time to increment the content of the MB.

The device interface logic shown in Figure 41 samples the normal Data Bit output signal for the most significant bit of the data word (BMB0) to determine if it overflows when incremented. If MB0 changes from the 0 to the 1 state when the data word is incremented, this logic requests a program interrupt to allow the program to take some appropriate action, such as incrementing a core memory counter for numbers above 4096, stopping the test to compile the data gathered to the current point in the operation, reinitializing the addressing, etc. The logic in the figure uses the select code of programmed data transfer operation to skip on the overflow condition to determine the cause of a program interrupt, to clear the overflow flip-flop, and to clear the device flag. Note that the devices that use data break transfers almost always use programmed data transfers to start and stop operation of the device, to initialize registers, etc., and do not rely on data break facilities alone to control their operations.


DEVICE
INTERFACE
PDP-8
IOT4
RESET
DEVICE
AND FLAG
PA
T2
SKIP

# Three-Cycle Data Breaks

Timing of input or output 3-cycle data breaks is shown in Figure 42. The 3-cycle break uses the block transfer control circuits of the computer. The block transfer control provides an economical method of controlling the flow of data at high speeds between PDP-8 core memory and fast peripheral devices, e.g., drum, disc, magnetic tape and line printers, allowing transfer rates in excess of 220 kc.

The three-cycle data break facility provides separate current address and word count registers in core memory for the connected device, thus eliminating the necessity for flip-flop registers in the device control. When several devices are connected to this facility, each is assigned a different set of core locations for word count and current address, allowing interlaced operations of all devices as long as their combined rate does not exceed 220 kc. The device specifies the location of these registers in core memory, and thus the software remains the same regardless of what other equipment is connected to the machine. Since these registers are located in standard memory, they may be loaded and unloaded directly without the use of IOT pulses. In a procedure where a device requests to transfer data to or from core memory, the three-cycle data break facility performs the following sequence of operations:

a. An address is read from the device to indicate the location of the word count register. This address is always the same for a given device; thus it can be wired in and does not require a flip-flop register.

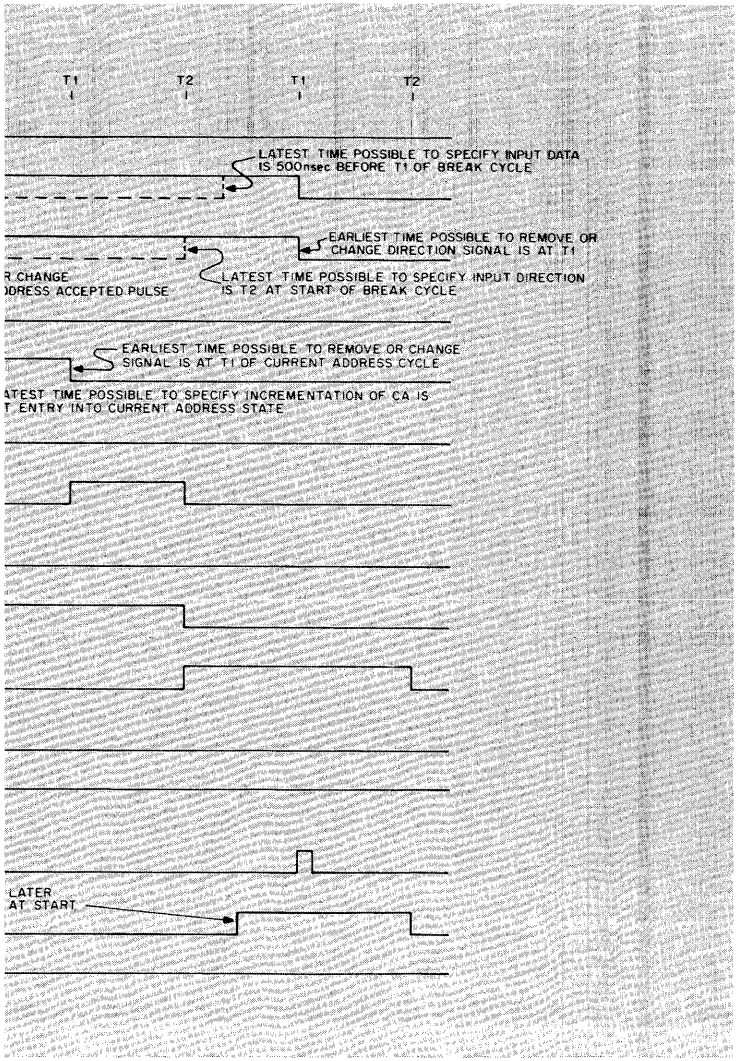
b. The content of the specified address is read from memory and 1 is added to it before rewriting. If the content of this register becomes 0 as a result of the addition, a WC Overflow pulse will be transmitted to the device. To transfer a block of N words, this register is loaded with − N during programmed initialization of the device. After the block has been fully transferred this pulse is generated to signify completion of the operation.

c. The next sequential location is read from memory as the current address register. Although the content of this register is normally incremented before being rewritten, an Increment CA Inhibit (+1 ⟶ CA Inhibit) signal from the device may inhibit incrementation. To transfer a block of data beginning at location A, this register is program initialized by loading with A-1.

d. The content of the previously read current address is transferred to the MA to serve as the address for the data transfer. This transfer may go in either direction in a manner identical to the single-cycle data break system.

The three-cycle data break facility uses many of the gates and transfer paths of the single-cycle data break system, but does not preclude the use of standard data break devices. Any combination of three-cycle and single-cycle data break devices can be used in one system, as long as a multiplexer channel is available for each. Two additional control lines are provided with the three-cycle data break. These are:

a. Word Count Overflow. A standard 0.4-μsec negative computer output pulse is transmitted to the device when the word count becomes equal to zero.




: Timing Diagram

b. Increment CA Inhibit. When ground potential, this device-supplied signal inhibits incrementation of the current address word.

In summary, the three-cycle data break is entered similarly to the single-cycle data break, with the exception of supplying a ground-level Cycle Select signal to allow entry of the WC (Word Count) state to increment the fixed core memory location containing the word count. The device requesting the break supplies this address as in the one-cycle data break, except that this address is fixed and can be supplied by wired ground and −3v signals, rather than from a register. The sole restriction on this address is that it must be an even number (bit 11 = 0). Following the WC state a CA (Current Address) state is entered in which the core memory location following the WC address (bit 11 = 1 after PC +1 ⇒ PC) is read, incremented by one, restored to memory, and used as the transfer address (by MB ⇒ MA). Then the normal B (Break) state is entered to effect the transfer.

# CHAPTER 4
# DIGITAL LOGIC CIRCUITS

PDP-8 is constructed of Digital FLIP CHIP modules. The Digital Logic Handbook, C-105, describes more than 100 of these modules, all their component circuits, and the associated accessories, i.e., power supplies and mounting panels. The user should study this catalog carefully before beginning the design of a special interface.

## Basic Digital Circuits

The basic component circuits used in the interface of the PDP-8 as well as in most FLIP CHIP modules are inverters, diode gates, diode-capacitor-diode (DCD) gates, pulse amplifiers, and bus drivers.

### INVERTERS

An inverter circuit is analogous to a switch. Figure 43 shows the basic inverter circuit. If the inverter base is at −3v and the emitter is at ground (most transistors in FLIP CHIP modules have a permanent connection from the emitter to ground), the PNP transistor is saturated and a conduction path is established between the emitter and collector output. Conversely, when the input at the base of the inverter is at ground, the transistor is cut off and the output at the collector goes negative. Collector points can connect to a load within the module or a remote load at the driven circuit. Internal collector loads are clamped at −3v. In series-R modules the load resistor is 7.5K to draw 2 ma, and in series-S modules the load resistor is 3K to draw 5 ma.

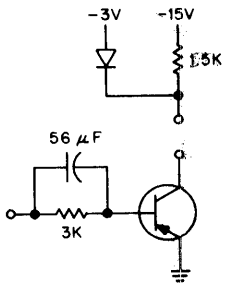



Figure 43 Inverter Circuit Schematic Diagram

Flip-flops are cross-coupled inverters using the same circuit. The state of a flip-flop is changed by driving the base of the conducting transistor to ground, thereby turning it off. Figure 44 shows the direct-set input circuit of the Type R210 PDP-8 Accumulator module.

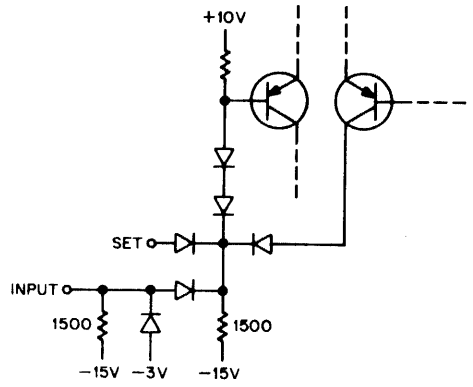



Figure 44 Direct-Set Input Circuit Schematic of the R210 PDP-8 Accumulator

## DIODE GATES

The diode gate is used in the R and S series to combine, amplify, invert, and standardize the signals which represent various logic functions. Figure 45 is a circuit schematic diagram of a simple diode gate with one input.

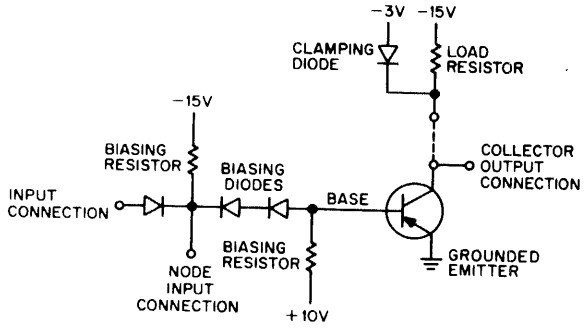



Figure 45 Single-Input Diode Gate Circuit Schematic

When the input is negative, the node point is also negative and current flows from the transistor emitter through the biasing diodes and the biasing resistor to −15v. As a result, the PNP transistor is turned on forming a short circuit between the collector and the emitter. Thus, when the input voltage is negative, the output voltage is ground potential. Since the output is from a saturated transistor, it has a low output impedance and good driving power.

When the diode gate input voltage is ground, the biasing diodes and the resistor, which is connected to the +10v supply, hold the transistor base more positive than the emitter, and the transistor is turned off. The output is then an open circuit, and it follows the voltage of any other circuit connected to it.

If the load resistor and clamp diode are attached to the transistor collector, they serve as a voltage source and hold the output at −3v while the transistor is off. When the transistor is on, the diode is cut off and the load resistor follows the output to ground.

The single-input diode gate therefore has three functions:

a. It inverts the input signal.

b. It standardizes the output voltage to −3v or ground (if the clamped load diode and resistor are connected).

c. Since the output current available from the transistor is much greater than the required input current, the diode gate amplifies.

A fourth function, gating, obtained by adding more diode inputs to the node point, is illustrated in Figure 46.

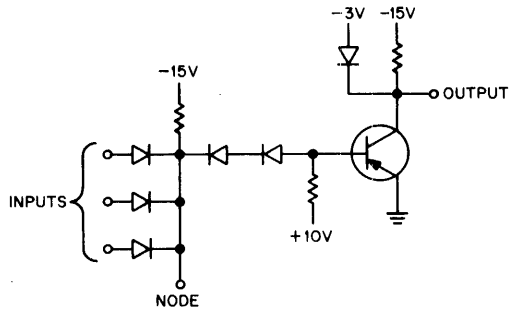



Figure 46 Multiple-Input Diode Gate Circuit Schematic

The node terminal is at approximately the same voltage as the most positive input. Thus, when any input terminal is grounded, the node terminal is also at ground and the circuit output is at −3v. If all of the inputs are negative, the node terminal is negative and the circuit output is at ground.

Figure 47 shows how gating functions can be performed by wiring together two or more diode gate outputs and one load resistor. When any input is negative, it saturates the corresponding transistor and forces the output line to ground. If all inputs are at ground, all of the transistors are open circuits and the output voltage, determined by the clamped load resistor, is −3v.

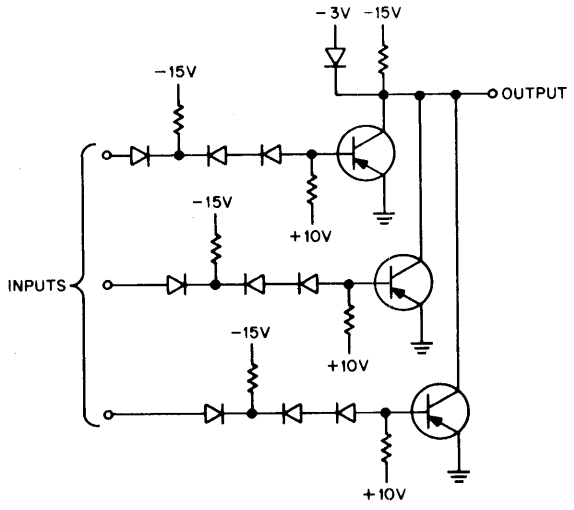



Figure 47 Parallel-Connected Diode Gate Circuit Schematic

It is possible to use the basic diode gate to construct very complex logical functions. A drawing showing all of the circuit components, however, would be difficult both to draw and read, so for this reason logic diagrams use a shorthand notation, representing one or more components as a single functional unit. Figure 48 shows a diode gate in the conventional way. The transistor circuit, including the biasing resistors and diodes, appears as a simple rectangle with an arrowhead indicating the direction of the transistor emitter. The load resistor appears as a resistor with a large dot at the top indicating that it is diode clamped to −3v.

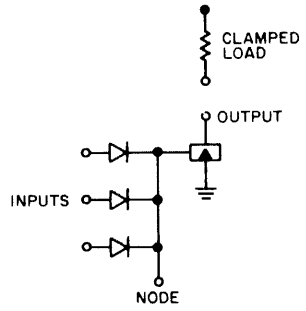



Figure 48 Diode Gate Logic Symbol

Diamonds show assertion input and output voltage levels. A solid diamond indicates a −3v level, and an open diamond indicates a ground level. In the 2-input diode gate of Figure 49, for example, if input A and input B are both negative, the output is at ground. If either A or B is at ground, the output is negative.

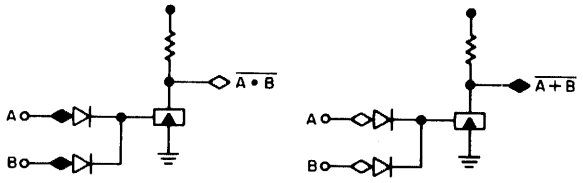



Figure 49 Logic Operations Performed by Diode Gates

## DIODE-CAPACITOR-DIODE GATES

The diode-capacitor-diode (DCD) gate is used to standardize the input to various units such as flip-flops, delays, and pulse amplifiers. It provides logical isolation between pulse and level inputs and produces a logical delay which is essential for sampling flip-flops at the same time they are being changed. It also acts as a logical AND gate since both pulse and level inputs must meet certain requirements for a signal to appear at the output. Either positive pulses or positive-going level changes (both −3v to ground) may be used as the pulse input.

A schematic drawing of a DCD gate is shown in Figure 50. If the level input is held at ground and the pulse input is held at −3v, the capacitor becomes charged after the set-up time has passed. If the pulse input then suddenly goes to ground, a positive-going pulse appears at the output. There is delay at the level input, but the pulse input goes to the output without delay. Even if the level input changes simultaneously with a positive transition at the pulse input, the delay acts as a temporary memory: the pulse input is gated according to the level input that existed during the interval before the pulse.

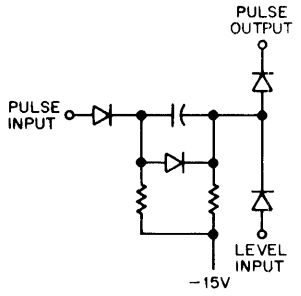



Figure 50 Diode-Capacitor-Diode Gate Circuit Schematic

An X in the rectangle distinguishes the symbol for the DCD gate (Figure 51) from the diode gate. The output is at the top, the delayed (level) input is at the bottom, and the differentiating (level change or pulse) input is on the side. An arrowhead rather than a diamond indicates the input signal to be differentiated, whether a level change or a pulse. The pulse symbols are hollow when positive-going and solid when negative-going. In the DCD gate, the pulse input must be positive-going.

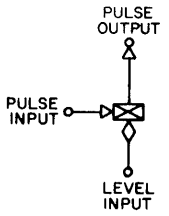



Figure 51 Diode-Capacitor-Diode Gate Logic Symbol

Since the same pulse may drive many DCD gates, the side of the rectangle opposite the pulse may be used to show a continuation of the same line, as in Figure 52. The illustration on the left below is a simplified version of the identical logical configuration on the right.

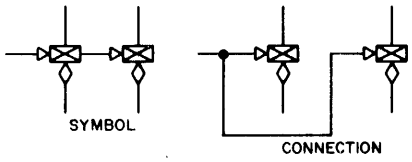



Figure 52 Parallel-Connected Trigger Pulse to DCD Gates

## PULSE AMPLIFIERS

The PDP-8 uses two types of pulse amplifier circuits. Modules such as the Type S603 contain monostable multivibrators (one-shots) to produce a standard 100-nsec negative output pulse. Modules such as the Type W640 use a transformer-coupled pulse-forming circuit to produce standard 400-nsec or 1-μsec pulses. The time required to saturate the inter-stage coupling transformer determines output pulse duration, and grounding the appropriate side of the output pulse transformer determines output polarity. Figure 53 shows schematically the final stages of this type of pulse amplifier circuit.

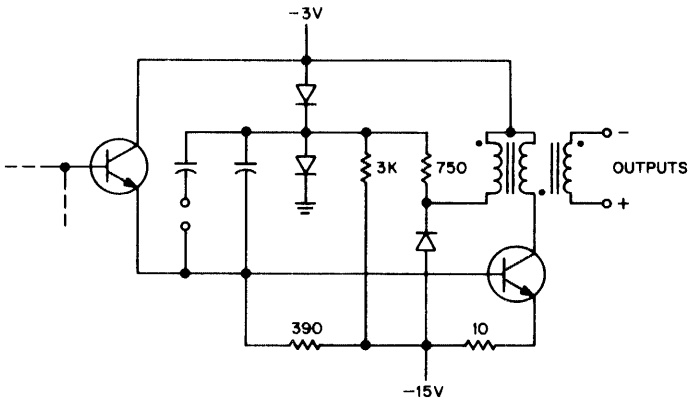



Figure 53 Pulse Amplifier Output Circuit Schematic

## BUS DRIVERS

Bus driver circuits that drive a heavy load contain a push-pull output stage, as shown in Figure 54. The Type R650 module uses a dc, inverting, amplifier circuit with a timing capacitor to control rise and fall times. With the capacitor shunted to ground, typical rise and fall time is 700 nsec; with this capacitor floating, typical rise and fall time is 50 nsec. A resistor output terminal is provided for driving coaxial cable.

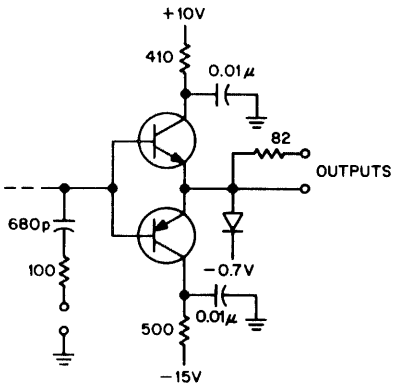



Figure 54 Bus Driver Output Circuit Schematic

# Interface Circuits of the Computer

Circuit modules of the PDP-8 receiving input signals and supplying output signals are as follows:

| Input | | Output | |
|---|---|---|---|
| Type | Name | Type | Name |
| A502 | Difference Amplifier | R650 | Bus Driver |
| R123 | Diode Gate | S107 | Inverter |
| R210 | PDP-8 Accumulator | W640 | Pulse Amplifier |
| R211 | MA, MB, and PC | | |
| S107 | Inverter | | |
| S111 | Diode Gate | | |
| S151 | Binary-to-Octal Decoder | | |
| S203 | Triple Flip-Flop | | |
| S603 | Pulse Amplifier | | |

## A502 DIFFERENCE AMPLIFIER

Only the PDP-8 containing a Analog-to-Digital Converter Type 189 option uses the A502 Difference Amplifier. The A502 is a high-speed difference amplifier which compares two input voltages and indicates which of the two is the more negative. The comparator has a resolution of 1 mv, and an input range of 0 to −10v. The maximum combined error due to a change in the common input voltage from 0 to −10v and a 20°C temperature change is 5 mv equivalent input offset. Two potentiometers allow adjustment of the zero set and common balance.

The comparator switching time is less than 250 nsec for a +10 mv square wave. The switching time is also less than 250 nsec when one input is at −5.00v and the other is switched from ground to −5.02v. For finer resolution, the switching time is increased. When the comparator is driven from a high impedance, fast switching source, such as a digital-to-analog converter, time should also be allowed for transients to settle.

The 0 to −10v input draws up to 1μa depending on the relative polarity of the two voltage inputs. The maximum current difference between positive and negative input voltages is 1 μa. The difference input capacitance is 75 pf.

## R123 DIODE GATE

The Data Line Interface Type 681 option uses the R123 as an interface module only where it receives the Line(1) Teletype signal. The R123 contains six 2-input negative NAND gates, with no load resistors for the gates since they usually drive the input of a flip-flop register. Standard ground and −3v levels with a duration of at least 100 nsec drive the input. Input load is 1 ma shared among the inputs that are at ground.

Standard ground and −3v levels are produced at the output. Each output can drive 20 ma of load at ground. The output terminals of diode gates may be connected in parallel. One clamped load is sufficient for parallel outputs when using less than 2 ft of wire. If the wire exceeds this length, additional clamped loads may be necessary for a sufficiently fast fall time in higher frequency applications. Two gates in parallel, driven by the same signal, can drive 38 ma at ground (20 ma each, less the 2-ma clamped load). Gates in parallel, not driven by the same signal, can drive 20 ma at ground minus 2 ma for each clamped load used.

## R210 PDP-8 ACCUMULATOR

The R210 is a double-height module that serves as one bit of the accumulator register, with all of the necessary input gates. Bus driver modules from this module apply outputs to the interface. Interface input connections consist of the direct-set connection used as the input bus for programmed data transfers. This input accepts standard levels of ground and −3v. A ground level of 400-nsec minimum duration activates the input. Input load is 11 ma at ground, and when not in use, the direct-set input terminal must be at −3v.

## R211 MA, MB, AND PC

The R211 is a double-height module that contains one bit of the memory address, memory buffer, and program counter registers of the PDP-8. Connections from this module to the interface receive the Data Address and Data Bit signals supplied by external equipment

that uses the data break facility. These signals are received as the level input to three DCD gates on each module. Two of these gates serve as a complementary input to the MA for the Data Address signal. An inverter in the module precedes the DCD gate on the clear side of the MA flip-flop; so the single address signal input either sets or clears the flip-flop. The third DCD gate connects to the 1 side of the MB flip-flop to receive the Data Bit signal directly. Control circuits within the computer provide trigger pulses to all these gates. Input signals must be at ground level to designate a binary 1 or −3v to designate a binary 0. These signals must precede the trigger pulse by at least 400 nsec. Each Data Address input represents 12 ma of load. Each Data Bit input represents 11 ma of load.

## R650 BUS DRIVER

The AC, MB, Break, and Run(1) output signals are buffered by bus driver circuits of Type R650 modules before connection to the interface.

The R650 contains two inverting bus drivers for driving heavy current loads to either ground or negative voltages. The bus drivers operate at frequencies up to 2 mc with typical rise and fall times of 25 nsec. The typical total transition times are 60 nsec for output rise and 65 nsec for output fall.

By grounding pin H or S the rise and fall time can be increased to avoid ringing on exceptionally long lines. The driver then operates at frequencies up to 500 kc with typical rise delay of 50 nsec, fall delay of 50 nsec, and total transition time of 800 nsec for output rise and 700 nsec for output fall. Terminal K or U can be used for driving coaxial cable.

The direct output (terminals J and T) drives 20 ma of external load at either ground or −3v. The resistor output (terminals K and U) drives 90-ohm coaxial cable such as RG-62-U. This output drives 5 ma of external load at either ground or −3v. The direct output connects to the interface connectors of the PDP-8.

## S107 INVERTER

In the basic computer, Type S107 Inverter modules receive the Increment MB and Cycle Select signals used with the data break. In the Memory Extension Control Type 183 option the S107 Inverter receives the Address Extension 1-3 signals, and in the Data Line Interface option Type 681 it receives the Teletype Line(1) signal. Within these two options the Type S107 supplies the Data Field and Teletype Instruction (TT Inst) output signals.

The S107 Inverter contains seven inverter circuits with single-input diode gates. Six of the circuits are used for single-input inversion; the seventh circuit can be used for gating by tying additional diode input networks to its node terminal. Clamped load resistors of 5 ma are a permanent part of each inverter. Typical output total transition times are 60 nsec for rise and 50 nsec for fall.

The diode input accepts standard level inputs of ground and −3v that are a minimum of 100 nsec in duration. This 1 ma input load is shared among the inputs at ground. The node terminal input accepts only R001 or R002 Diode Network output connections, or their equivalents. The combined length of all leads attached to the node terminal must not exceed 6 inches. Input signal and load characteristics for diode networks are the same as those given for the diode input above.

Output signals from the inverters are standard levels of ground and −3v. Each inverter drives 15 ma of load at ground. Output terminals of inverters may be connected in parallel. Only one clamped load resistor is needed at the output when less than 2 feet of wire is used. If the wire exceeds this length, additional clamped load resistors may be necessary for a fast enough fall time in high frequency applications.

## S111 DIODE GATE

Type S111 modules in the computer receive the Program Interrupt Request and Transfer Direction (Data In) signals.

The S111 Diode Gate contains three diode gates, each connected to a transistor inverter. The gate operates as a NAND for negative inputs and as a NOR for ground inputs. Each gate has three input terminals: two are connected to diodes; a third is connected directly to the node point of the diode gate. The third terminal allows the number of input diodes to be increased by adding external diode networks such as the R001 or R002 modules. External diodes must be connected in the same direction as the diodes in the S111. Typical output total transition times are 60 nsec for rise and 50 nsec for fall.

Input signals to the diode terminals must be standard levels of ground or −3v, and must have a minimum duration of 100 nsec. Input load is 1 ma shared among the inputs at ground. Input signals to the node terminals accept only connections from Type R001 or R002 Diode Network modules, or their equivalents. The maximum combined length of all leads attached to a node terminal is 6 inches. Input signal load is similar to the diode input.

## S151 BINARY-TO-OCTAL DECODER

A Type S151 module (in parallel with a Type S107) receives Address Extension 1-3 signals supplied to the Memory Extension control option Type 183.

This S151 decodes binary information from three flip-flops into octal form. When the enable input is at ground, the selected output line is at ground and the other seven outputs are at −3v. When the enable input is at −3v, all outputs are at −3v. The internal gates are similar to those in the S111. The enable input is the common emitter connection of the output inverters. Typical total transition times are 75 nsec for output rise and 60 nsec for output fall.

Standard input levels are −3v and ground, 100 nsec minimum duration. The 2.3 ma input load is shared among the inputs at ground.

## S203 TRIPLE FLIP-FLOP

The S203 contains three identical flip-flops. Each flip-flop has a direct clear and a DCD gate for conditional readin. The level input to one of these gates connects to the interface to receive the Data Break Request signal. This input receives standard ground and −3v levels. The conditioning level must be ground level to condition the gate and to request a break. This conditioning must occur at least 400 nsec before an internal computer pulse triggers the gate. The level input represents a 2-ma load at ground.

## S603 PULSE AMPLIFIER

Pulse amplifiers of Type S603 modules receive the Clear AC and Skip inputs of the PDP-8. An S603 module contains three pulse amplifier circuits for power amplification and standardizing pulses in amplitude and width. Each amplifier produces standard 100-nsec negative pulses each time the input triggers from the diode or DCD gate inputs. Both interface input signals flow to the diode input of a pulse amplifier. The pulse amplifier can accommodate input pulses at any frequency up to 2 mc. Delay through the pulse amplifier is approximately 50 nsec. The diode input receives standard 100-nsec pulses (−3v to ground) or positive-going level changes (−3v to ground) with a rise time no longer than 60 nsec. The input level must be returned to −3v for at least 400 nsec before another input may occur at either the diode or DCD gate input. The diode input represents a 1-ma load at ground.

## W640 PULSE AMPLIFIER

The IOP pulses, BT1 and BT2 timing pulses, and the B Power Clear pulses are supplied to the interface as outputs from Type W640 modules. The W640 module contains three ungated pulse amplifiers capable of producing either 1-μsec or 400-nsec pulses. In the normal PDP-8, these outputs are standard negative 400-nsec pulses. Each output drives 10 ma of load (equivalent to 10 inverter bases such as the B104 or S111). These amplifiers should not be used without a terminating resistor; typical values are 47 to 150 ohms.

# Interface Circuits of Peripheral Equipment

Several FLIP CHIP circuit modules are of particular interest in the design of equipment to interface with the PDP-8. Chief among these are the W103 Device Selector and the R123 Diode Gate. In addition to these, the following modules serve special interface applications:

| Type | Name | Application |
|---|---|---|
| B681 | Power Inverter | General purpose digital logic to supply or operate devices requiring up to 32 ma. |
| B684 | Bus Driver | Provides output driving current of up to 40 ma. |
| W040 | Solenoid Driver | Provides driving capability for electromechanical devices such as counters, clutches, and other solenoid-actuated equipment requiring currents of up to 600 ma at −2.5 to −70v. |
| W051 | Driver | Provides driving capability for devices that require up to 100 ma at 0 to −15v. |
| W501 | Schmitt Trigger | Provides level conversion from voltages of ±10v to the standard levels required by DEC modules. |
| W510 | Positive Input Converter | Converts positive voltage signal levels to ground and −3v levels for DEC modules. |

| Type | Name | Application |
|---|---|---|
| W600 | Negative Output Converter | Converts standard DEC logic levels of ground and −3v to levels of ground and a negative voltage between −1 and −15v determined by the level of an external supply. |
| W601 | Positive Output Converter | Converts standard DEC logic levels of ground and −3v to levels of ground and some positive voltage between +1 and +20v as determined by the level of an external supply. |

## W103 DEVICE SELECTOR

The W103 selects an input/output device according to the code in the instruction word (being held in the memory buffer during the IOT cycle). Figure 55 shows the logic circuits of the W103 module.

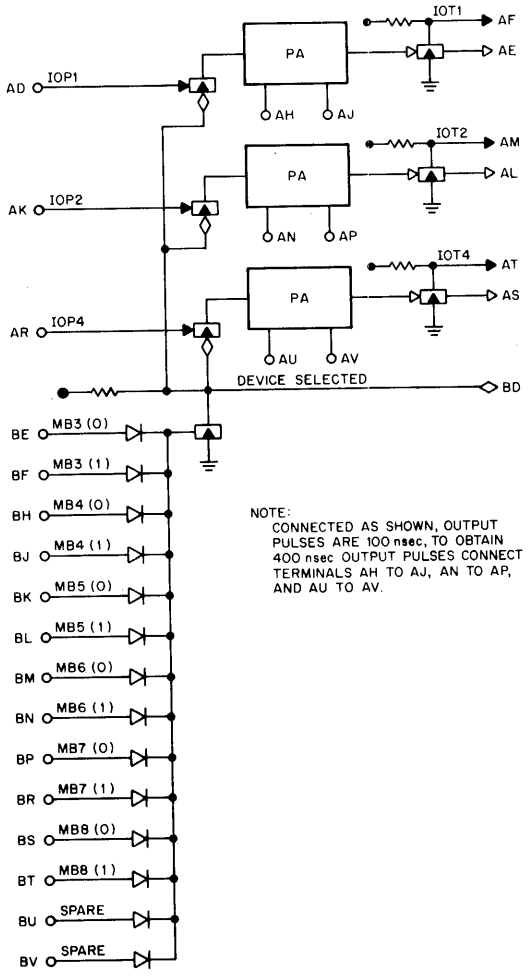


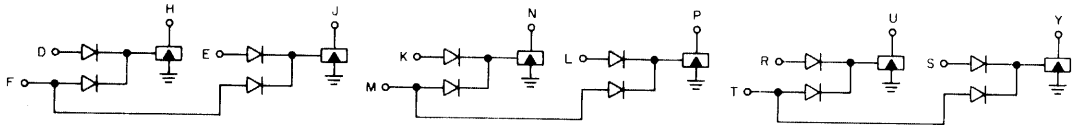

Figure 55 Device Selector W103 Logic Circuit

The twelve input diodes permit selection of any arbitrary 6-bit code, and decode the number held in MB bits 3 through 8. When the proper enabling code arrives at the diode gate, the three input gates driving the three pulse amplifiers are enabled and permit passage of the programmed IOP pulses. To establish a code on the module, the six unnecessary diodes are disabled by snipping one of their leads or removing them altogether. If MB bit 3 is a binary 1 to set up the correct code, the diode going to the binary 0 side of MB bit 3 is disabled. Two spare diodes are included for additional gating flexibility.

Three pulse amplifiers produce R-series 100-nsec positive-going output pulses. Inverted pulse amplifier output pulses are provided for gates (such as the Type R111 Diode Gate) which require negative or negative-going pulses. Jumper terminals on each pulse amplifier establish a pulse duration of 400 nsec for the output pulse. It is recommended that the 400-nsec pulse duration be used when transmitting the pulse over long distances. The 400-nsec pulse also clears R-series flip-flops whose carry gates are permanently enabled. The positive pulse output of each pulse amplifier is rated 65 ma of external load at ground; the negative output is rated 15 ma at ground (when driving a load connected to -15v). These outputs are not designed to drive loads when at -3v (loads connected to ground). To drive this type of load, a clamped load resistor must be connected to the pulse amplifier output terminal to supply the current.

## R123 DIODE GATE

This module contains six 2-input NAND gates for negative levels and is useful for transferring data into or out of the PDP-8 accumulator. Standard DEC negative levels or 0.4 microsecond negative pulses such as those from the W103 Device Selector can be used as input signals. Input load per gate is 1 ma shared among the inputs at ground.




NOTES:
1. STROBE PULSE INPUT TO TERMINALS F, M, AND T WHICH ARE CONNECTED IN COMMON WHEN USED AS A BUS GATE
2. DATA BIT INPUTS TO TERMINALS D, E, K, L, R, AND S
3. TWO MODULES ARE REQUIRED TO STROBE A 12-BIT WORD

Figure 56 Diode Gate R123 Logic Circuit

Two R123 modules provide sufficient gating to transfer one 12-bit word into the accumulator. If more gates are needed to load the AC from several sources, the output terminals can be OR connected by bussing together additional gate collectors.

# CHAPTER 5
# INTERFACE CONNECTIONS

All interface connections to the PDP-8 are made at assigned module receptacle connectors in the left (memory-M) or right (processor-P) mounting frame (door). Capital letters designate horizontal rows of modules within a mounting frame from top to bottom. Module receptacles are numbered from left to right as viewed from the wiring side (right to left from the module side). Terminals of a connector or module are assigned capital letters from top to bottom, omitting G, I, O, and Q. Therefore, terminal PE2H is in the right mounting frame (P), the fifth row from the top (E), the second module from the left (2), and the seventh terminal from the top of the connector (H).

The module receptacles and assigned assigned use for interface signal connections are:

| Receptacle | Signal Use |
|---|---|
| PE2 | AC 0-8 inputs |
| PE3 | Data Address 0-8 inputs |
| PE4 | Data Bit 0-8 inputs |
| PF2 | AC 9-11, Skip, Clear AC inputs and Run output |
| PF3 | Data Address 9-11 inputs, and Address Accepted and B Break outputs |
| PF4 | Data Bit 9-11 inputs |
| ME30 | Address Extend 1, 2, 3 inputs and Data Field 0-2 outputs |
| ME34 | BAC 0-8 outputs |
| ME35 | BMB 0-5 outputs |
| MF34 | BAC 9-11, IOTs, BT1, BT2A, and B Power Clear outputs |
| MF35 | BMB 6-11 outputs |

Terminals C, F, J, L, N, R, and U of these receptacles are grounded within the computer and terminals D, E, H, K, M, P, S, T, and V carry signals. These terminals mate with Type W011 Signal Cable Connectors at each end of 93-ohm coaxial cable.

Interface connection to the PDP-8 can be established for all peripheral equipment by making series cable connections between devices. In this manner only one set of cables is connected to the computer and two sets are connected to each device: one receiving the computer connection from the computer itself or the previous device; and one passing the connection to the next device. Where physical location of equipment does not make series bus connections feasible, or when cable length becomes excessive, additional interface connectors can be provided near the computer.

All logic signals passing between the PDP-8 and the input/output equipment are standard DEC levels or standard DEC pulses. Logic signals have mnemonic names that indicate the condition represented by assertion of the signal. Standard levels are either ground potential (0.0 to −0.3v), designated by an open diamond (——◇) or are −3v (−3.0 to −4.0v), designated by a solid diamond (——◆). Standard pulses in the positive direction are designated by an open triangle (——▷) and negative pulses are designated by a solid triangle (——▶). Pulses originating in R or S series modules are positive-going pulses which start at −3v, go to ground for 100 nsec, then return to −3v. Pulses originating in W series modules are always negative, are always referenced to ground, are 2.5v in amplitude (2.3 to 3.0v) with a 2v overshoot, and are of 400-nsec duration.

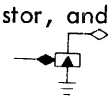
The following tables present cable connections and logic circuit identification information for PDP-8 interface signals. Computer input signals that must drive the interface bus to ground (data inputs to the AC, Clear AC, Skip, and Interrupt Request) must be connected to the collector of a grounded-emitter transistor, and so can be considered transistor-gated negative pulses (  ) or levels (  ).

TABLE 5 PROGRAMMED DATA TRANSFER INPUT SIGNALS

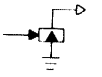
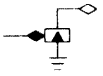
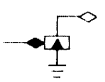
| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| AC 0 |  | PE2D | PA7E | R210 |
| AC 1 |  | PE2E | PA8E | R210 |
| AC2 |  | PE2H | PA9E | R210 |
| AC 3 |  | PE2K | PA10E | R210 |
| AC 4 |  | PE2M | PA11E | R210 |
| AC 5 |  | PE2P | PA12E | R210 |
| AC 6 |  | PE2S | PA13E | R210 |
| AC 7 |  | PE2T | PA14E | R210 |
| AC 8 |  | PE2V | PA15E | R210 |
| AC 9 |  | PF2D | PA16E | R210 |
| AC 10 |  | PF2E | PA17E | R210 |
| AC 11 |  | PF2H | PA18E | R210 |
| Clear AC |  | PF2P | PA19J | S603 |
| Interrupt Request |  | PF2M | PD36K | S111 |
| Skip |  | PF2K | PB21V | S603 |

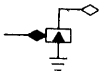
TABLE 6 PROGRAMMED DATA TRANSFER OUTPUT SIGNALS

| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| BAC 0 (1) | —◇ | ME34D | ME26J | R650 |
| BAC 1 (1) | —◇ | ME34E | ME26T | R650 |
| BAC 2 (1) | —◇ | ME34H | ME27J | R650 |
| BAC 3 (1) | —◇ | ME34K | ME27T | R650 |
| BAC 4 (1) | —◇ | ME34M | ME28J | R650 |
| BAC 5 (1) | —◇ | ME34P | ME28T | R650 |
| BAC 6 (1) | —◇ | ME34S | MF26J | R650 |
| BAC 7 (1) | —◇ | ME34T | MF26T | R650 |
| BAC 8 (1) | —◇ | ME34V | MF27J | R650 |
| BAC 9 (1) | —◇ | MF34D | MF27T | R650 |
| BAC 10 (1) | —◇ | MF34E | MF28J | R650 |
| BAC 11 (1) | —◇ | MF34H | MF28T | R650 |
| IOP 1 | —▶ | MF34K | MC31H | W640 |
| IOP 2 | —▶ | MF34M | MC31N | W640 |
| IOP 4 | —▶ | MF34P | MC31U | W640 |
| BMB 3 (0) | —◇ | ME35K | MC27T | R650 |
| BMB 3 (1) | —◇ | ME35M | MC28J | R650 |
| BMB 4 (0) | —◇ | ME35P | MC28T | R650 |
| BMB 4 (1) | —◇ | ME35S | MC29J | R650 |
| BMB 5 (0) | —◇ | ME35T | MC29T | R650 |
| BMB 5 (1) | —◇ | ME35V | MD25J | R650 |
| BMB 6 (0) | —◇ | MF35D | MD25T | R650 |
| BMB 6 (1) | —◇ | MF35E | MD26J | R650 |
| BMB 7 (0) | —◇ | MF35H | MD26T | R650 |
| BMB 7 (1) | —◇ | MF35K | MD27J | R650 |
| BMB 8 (0) | —◇ | MF35M | MD27T | R650 |
| BMB 8 (1) | —◇ | MF35P | MD28J | R650 |

TABLE 7 DATA BREAK TRANSFER INPUT SIGNALS

| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| Data Address 0 (1) | —◇ | PE3D | PC7R | R211 |
| Data Address 1 (1) | —◇ | PE3E | PC8R | R211 |
| Data Address 2 (1) | —◇ | PE3H | PC9R | R211 |
| Data Address 3 (1) | —◇ | PE3K | PC10R | R211 |
| Data Address 4 (1) | —◇ | PE3M | PC11R | R211 |
| Data Address 5 (1) | —◇ | PE3P | PC12R | R211 |
| Data Address 6 (1) | —◇ | PE3S | PC13R | R211 |
| Data Address 7 (1) | —◇ | PE3T | PC14R | R211 |
| Data Address 8 (1) | —◇ | PE3V | PC15R | R211 |
| Data Address 9 (1) | —◇ | PF3D | PC16R | R211 |
| Data Address 10 (1) | —◇ | PF3E | PC17R | R211 |
| Data Address 11 (1) | —◇ | PF3H | PC18R | R211 |
| Data Bit 0 (1) | —◇ | PE4D | PD7M | R211 |
| Data Bit 1 (1) | —◇ | PE4E | PD8M | R211 |
| Data Bit 2 (1) | —◇ | PE4H | PD9M | R211 |
| Data Bit 3 (1) | —◇ | PE4K | PD10M | R211 |
| Data Bit 4 (1) | —◇ | PE4M | PD11M | R211 |
| Data Bit 5 (1) | —◇ | PE4P | PD12M | R211 |
| Data Bit 6 (1) | —◇ | PE4S | PD13M | R211 |
| Data Bit 7 (1) | —◇ | PE4T | PD14M | R211 |

TABLE 7 DATA BREAK TRANSFER INPUT SIGNALS (continued)

| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| Data Bit 8 (1) |  | PE4V | PD15M | R211 |
| Data Bit 9 (1) |  | PF4D | PD16M | R211 |
| Data Bit 10 (1) |  | PF4E | PD17M | R211 |
| Data Bit 11 (1) |  | PF4H | PD18M | R211 |
| Break Request |  | PF3K | PC32J | S203 |
| Transfer Direction |  | PF3M | PD23E | S111 |
| Increment MB |  | PF3T | PD31M | S107 |
| Cycle Select |  | PF4K | PE7S | S107 |
| Increment CA |  | PF4M | PE10F | R121 |

*Direction is into PDP-8 when signal is −3v, out of PDP-8 when ground potential.

**The Increment MB input to the PDP-8 must be the output of a gating circuit that enables generation of the ground level signal only when the B Break signal is present.

TABLE 8 DATA BREAK TRANSFER OUTPUT SIGNALS

| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| BMB 0 (1) |  | ME35D | MC26J | R650 |
| BMB 1 (1) |  | ME35E | MC26T | R650 |
| BMB 2 (1) |  | ME35H | MC27J | R650 |
| BMB 3 (1) |  | ME35M | MC28J | R650 |
| BMB 4 (1) |  | ME35S | MC29J | R650 |
| BMB 5 (1) |  | ME35V | MD25J | R650 |
| BMB 6 (1) |  | MF35E | MD26J | R650 |

TABLE 8 DATA BREAK TRANSFER OUTPUT SIGNALS (continued)

| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| BMB 7 (1) | | MF35K | MD27J | R650 |
| BMB 8 (1) | | MF35P | MD28J | R650 |
| BMB 9 (1) | | MF35S | MD28T | R650 |
| BMB 10 (1) | | MF35T | MD29J | R650 |
| BMB 11 (1) | | MF35V | MD29T | R650 |
| B Break | | PF3P | PE8T | R650 |
| Address Accepted | | PF3S | PF10H | W640 |
| WC Overflow | | PF4P | PF10N | W640 |

TABLE 9 MISCELLANEOUS INPUT SIGNALS

| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| ADDR Extension 1 | | ME30D | ME8K, MC3K | S107, S151 |
| ADDR Extension 2 | | ME30E | ME8H, MC3E | S107, S151 |
| ADDR Extension 3 | | ME30H | ME8E, MC3J | S107, S151 |
| Analog In* | | Special BNC | PE11N | A502 |

*Input to Analog-to-Digital Converter Type 189 is made to BNC connector on back of processor fan mounting.

TABLE 10 MISCELLANEOUS OUTPUT SIGNALS

| Signal | Symbol | Interface Connection | Module Terminal | Module Type |
|---|---|---|---|---|
| B Run (1) | | PF2S | PE8J | R650 |
| Data Field 0 (1) | | ME30K | ME7L | S107 |
| Data Field 1 (1) | | ME30M | ME7N | S107 |
| Data Field 2 (1) | | ME30P | ME7R | S107 |
| BT1 | | MF34S | MD30H | W640 |
| BT2A | | MF34T | MD30U | W640 |
| B Power Clear | | MF34V | MD30N | W640 |

# Miscellaneous Interface Signals

The following input and output signal connections are available for use with DEC equipment options or for use in special interface equipment designed by the customer.

## ADDRESS EXTENSION INPUTS AND DATA FIELD OUTPUTS

When the Memory Extension Control Type 183 is in the computer system, devices using the data break facility must supply a 12-bit data address and a 3-bit address extension. Conversely, the programmed transfer of an address to a register in a device that uses the data break occurs as a 12-bit word from the accumulator and a 3-bit data field extension from the 183.

The Address Extension 1-3 signals must be ground potential to designate a binary 1 and −3v to designate a binary 0. Each of these signals supplies an input to both an inverter of a Type S107 module and a Type S151 Binary-to-Octal Decoder module. Each signal at ground potential is loaded by 2 ma and each signal at −3v receives no load.

The Data Field 0-2 signals are constantly available at the interface connectors. They are flip-flop output signals buffered by an inverter of a Type S107 module. Each signal can drive 15 ma at ground potential, specifying a binary 1.

## ANALOG INPUT SIGNAL

The Analog-to-Digital Converter Type 189 option receives an analog input signal between 0 and −10v. A BNC connector mounted on the outside of the processor fan housing at the back of the computer provides connection for this signal. Internal wiring cables this connector to the input of a Type A502 Comparator module. This module compares the analog input signal with a 0 to −10v analog signal produced in Type A601 and A604 Digital-Analog Converter modules. The input draws up to 1 μa depending on the relative polarity of the two voltage inputs of the A502 module. The maximum current difference between positive and negative input voltages is 1 μa. The difference input capacitance is 75 pf.

## B RUN OUTPUT SIGNAL

The binary 1 output of the RUN flip-flop flows to external equipment through the interface circuits. This signal is at −3v when the computer is performing instructions and is at ground potential when the program halts. Magnetic tape and DECtape equipment use this signal to stop transport motion when the PDP-8 halts, preventing the tape from running off the end of the reel. The B Run signal is routed to the interface connector through a Type R650 Bus Driver module which can drive a 20-ma load.

## BT1 AND BT2A OUTPUT PULSES

Two buffered timing pulse signals, designated BT1 and BT2A, are supplied to I/O devices. These signals can synchronize operations in external equipment with those in the computer. The BT1 and BT2A pulse signals are derived from the T1 and T2A pulse signals generated by the timing signal generator of the PDP-8. The Type W640 Pulse Amplifier module

standardizes the T1 and T2A pulses as negative 400-nsec pulses. The resulting (buffered) BT1 and BT2A pulses are supplied to the interface connections. Interface cable connections for each of the pulse outputs can drive a 10-ma load.

### B POWER CLEAR OUTPUT PULSES

The Power Clear pulses generated and used within the PDP-8 are made available at the interface connections. External equipment uses these pulses to clear registers and control logic during the power turn-on period. Use of Power Clear pulses in this manner is valid only when the logic circuits cleared by the pulses are energized before or at the same time the PDP-8 POWER switch is turned on.

## Loading and Driving Considerations

All PDP-8 circuits providing output or receiving input interface signals are R-, S, or W-series FLIP CHIP modules. Therefore the PDP-8 interface is defined entirely in terms of current driving or draining characteristics.

All R- and S-series modules are capable of driving currents in the direction from ground to −15v, assuming Ben Franklin's definition of current flow. In no cases are R- or S-series modules designed to drive loads which are essentially base loads. If such loads are to be driven, extra clamped load resistors must be added to make up the necessary differential in current. Therefore, R- and S-series loads are defined in terms of milliamperes at ground and 0 ma at −3v. In general, the output of any R- or S-series inverter, including the flip-flop, can drive 20 ma. However, a 2-ma load in R or 5-ma load in S modules is included within most flip-flops and this current must be deducted from the available driving capability.

Inputs are also defined in terms of milliamperes. Level inputs to level gates are defined as 1 ma at ground. Level inputs to diode-capacitor-diode (DCD) gates are 2 ma at ground, and pulse inputs to DCD gates are 3 ma at ground. Since capacitive loading presents problems with R-series modules, where long lines are being driven, the user should add extra clamped loads to sufficiently discharge cable capacitance. Approximately an extra 2 ma of clamped load current should be added for every foot of wire beyond 1-1/2 ft. Inputs to the Type R650 Bus Driver module are exempted from this rule since this module is designed to drive coaxial cable of 93-ohms characteristic impedance.

The Type R650 Bus Driver module has two types of outputs, the fast and the slow (or ramp) output. Using the fast output, the bus driver operates as a fast amplifier. In using the ramp output, an integrating capacitor between input of the bus driver and the output stage, causes the output lines to move from ground to −3v or in the reverse direction in approximately 500 nsec. This connection on the AC lines reduces cross-talk between lines. All other R650 module outputs are fast.

The Type W640 Pulse Amplifier modules should be carefully terminated. If sufficient noise is generated at the output of the W640, it may cause the pulse amplifier to regenerate; hence it is also recommended that output lines of W640 modules be well shielded. The outputs of W640 modules may be either 400 nsec or 1 μsec in width. All connections on the standard PDP-8 use the 400-nsec pulse width.

Input signals to the PDP-8 are, in many cases, a clamped load resistor of 10 ma and a direct input to a flip-flop or pulse amplifier. The input load is, therefore, 10 ma for the

clamped load and 1 ma for the flip-flop or the pulse amplifier. Capacitive loads must be at −3v before the pulse amplifier or flip-flop is used for the next machine cycle. There are some exceptions to this statement. First, the Data Bit inputs to the MB require 2 ma at ground and no current at −3v. The Transfer Direction signal requires 1 ma at ground. The Break Request signal input has a 10-ma clamped load plus 2 ma for the internal circuitry or 12 ma at ground. The output of all DEC inverters in the system module series drives 15 ma. In general, the clamped load which is normally used absorbs 10 ma. However, on the input signal interface, no clamped load is used; hence, the above numbers may be used.

Timing is, in general, determined by the machine itself. However, a few statements can be made about module timing. The Type S111 Diode Gates set up in approximately 50 nsec in either direction under normal load conditions. Fall times are faster with heavier loads. The diode-capacitor-diode gates set up in 400 nsec. This 400 nsec is determined from the end of the preceding 100-nsec pulse, and both the level and pulse must return to −3v for 400 nsec before the next pulse arrives. Pulses originating in R- or S-series modules are 100 nsec in width, measured from the 10% point of the leading edge to the 90% point of the trailing edge. Fall time is not critical on these pulses provided the pulse has returned to −3v in time to come up for the next pulse.

The following definitions and rules serve as a useful guide in determining the driving capability of output signals and the load presented to input signals by B-series FLIP CHIP modules or DEC System Modules used in peripheral equipment connected to the PDP-8.

## BASE LOAD

Base load is the current which must be drawn from the base of a dc inverter to keep it saturated. In this condition the inverter circuit input terminal is at −3v, the emitter is at ground, and a nominal 1 ma of current flows through the 3000-ohm base resistor from ground. A 1500-ohm load resistor clamped at −3v can nominally accept 8 ma, but tolerance considerations limit this number to 7 ma. Thus, an inverter collector with a 1500-ohm clamped load can drive a maximum of seven base loads.

## PULSE LOAD

Pulse load is the load presented to the output of a pulse source by an inverter base in the same speed series, or by the direct set or clear input of a flip-flop. Pulse amplifiers are usually limited to driving 16 pulse loads. This number should be decreased if the bases are widely separated physically, and can be increased to 18 if the bases are physically close together. The series inductance and shunt capacity of connecting wires make pulses at the end of a series of bases either large or small. Consequently, when driving nearly the maximum number of bases, the pulse amplitude should be carefully checked after installation. A terminating resistor in the 100- to 300-ohm range is desirable to reduce ringing on a heavily loaded pulse line. The loading on a pulse source is approximately the same when driving a base as a direct input to a flip-flop. One pulse source, of course, cannot drive both direct input of flip-flops and inverter bases because the direct inputs require DEC standard positive pulses, and base inputs require DEC standard negative pulses. A pulse load is largely determined by the value of the speed-up capacitor connected in parallel with the 3000-ohm base resistor. In the 4000-series 500-kc modules this capacitor is 680 pf; in 1000-series 5-mc modules it is 82 pf; and in 6000-series 10-kc modules it is 56 pf.

## PULSED EMITTER LOAD

Pulsed emitter load is the load applied to the collector of an inverter which drives the pulse input to a flip-flop, pulse amplifier, or delay. The pulse current passes through all of the inverters in series with the pulse input, and it should be assumed to be the load on each of the series inverters.

## DC EMITTER LOAD

The load applied to the collector of an inverter driving a clamped load resistor is the dc emitter load. This load is also presented by the collector of an inverter which drives an emitter in an inverter network terminated by a clamped load resistor. Under these conditions, the collector of an inverter driving an emitter in a transistor gating network must also supply the base current leaving the succeeding inverters which are saturated. This current is small, but in complex networks it must be considered. An inverter in the DEC 1000- or 6000-series modules can supply 15 ma, and in the 4000-series modules can supply 20 ma.

An inverter network can always be analyzed by assuming:

1. A short circuit exists between the emitter and collector when −3v is applied to the base.

2. Base current of 1 ma will flow if either the collector or emitter is held at ground potential.

3. The maximum dc collector current through an inverter is 20 ma for 4000-series 500-kc modules and is 15 ma for all other DEC series modules.

A capacitor-diode gate level input does not present any dc load. A transient load occurs when the input level changes. Note that all capacitor-diode gates require that the level input precede the initiating pulse input by at least 1 μsec.

# CHAPTER 6

# INSTALLATION PLANNING

## Space Requirements

Space must be provided at the installation site to accommodate the PDP-8 and peripheral equipment and to allow access to all doors and panels for maintenance.

Installation dimensions for a table-mounted and a rack-mounted PDP-8 are shown in Figures 57 and 58, respectively. Dimensions of a rack-mounted PDP-8 in an optional DEC computer cabinet and of a table-mounted PDP-8 on an optional DEC winged table are shown in Figure 59. Floor space for a basic optional computer cabinet is 22-1/4 inches wide (with two end panels) and 27-1/16 inches deep, plus additional space for a table. Figure 59 can be used in planning the installation of all I/O equipment mounted in standard computer cabinets noting that other cabinets may be equipped with a table, and that cabinets bolted together are 19-3/4 inches wide with 1-1/4 inch end panels mounted on the outer ends. Minimum service clearance on all standard DEC computer cabinets is 8-3/4 inches at the front and 14-7/8 inches at the back. A standard DEC computer cabinet contains space for one mounting panel (two rows) of FLIP CHIP modules or an indicator panel above the computer, and for three mounting panels below the operator console. The memory frame and the processor frame are hinged to provide access to the wiring side of the mounting panels. Both of these frames extend beyond the back of the power supply in the table model to allow entrance of interface cables. Cables enter the cabinet model through a port in the bottom of standard cabinets. Wheels and leveling devices on the cabinets allow cable clearance so that subflooring is not required.

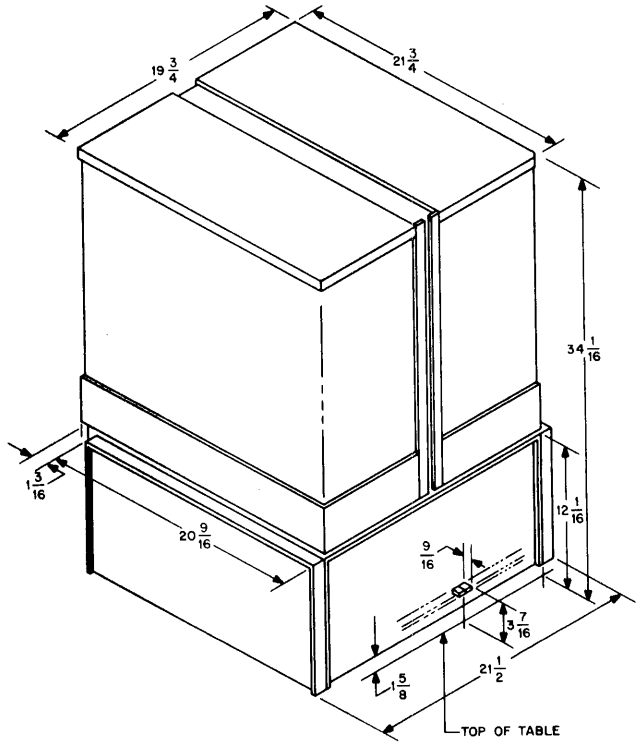



Figure 57 Table Mounted PDP-8 Installation Dimensions

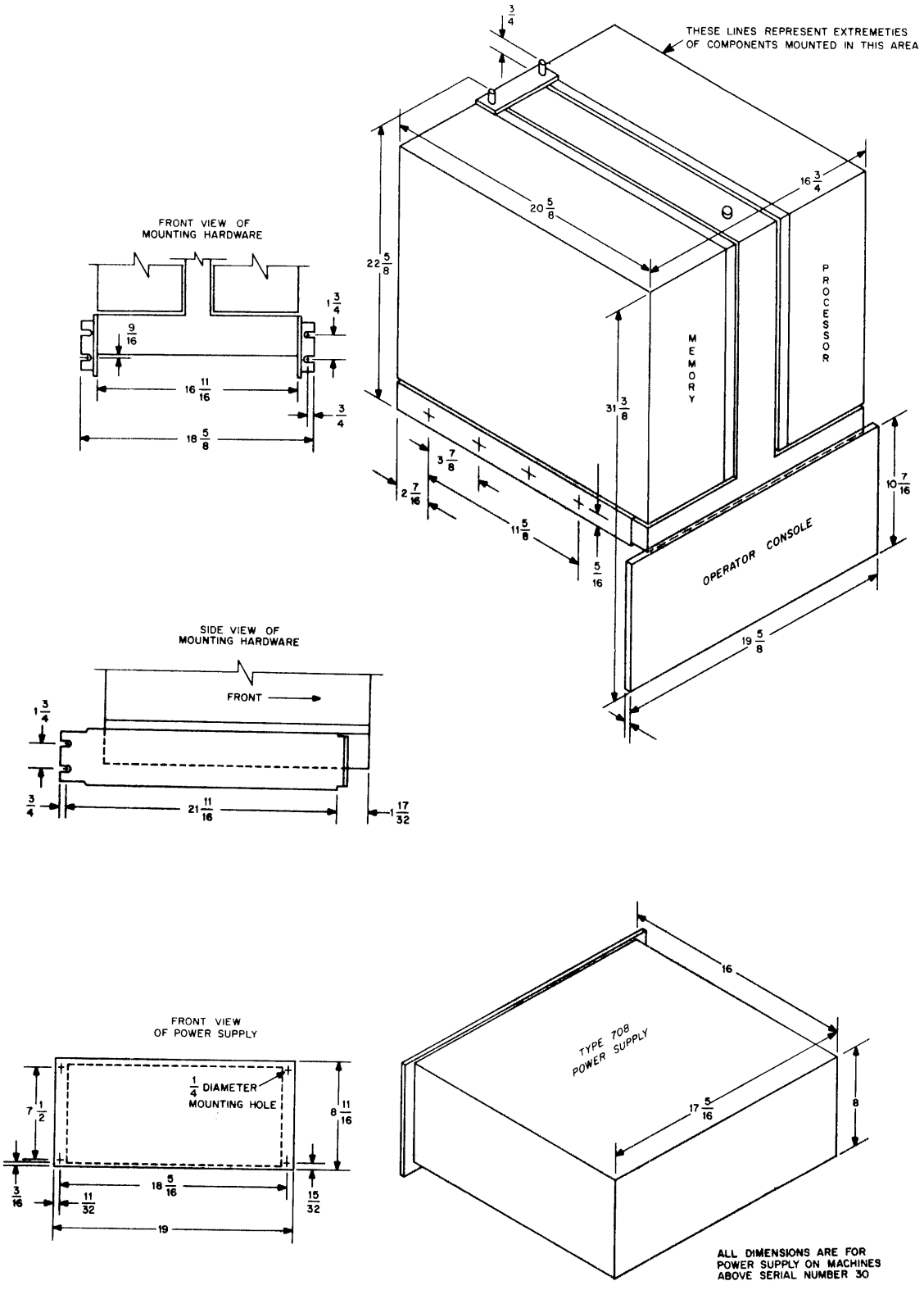



Figure 58 Cabinet Mounted PDP-8 Installation Dimensions

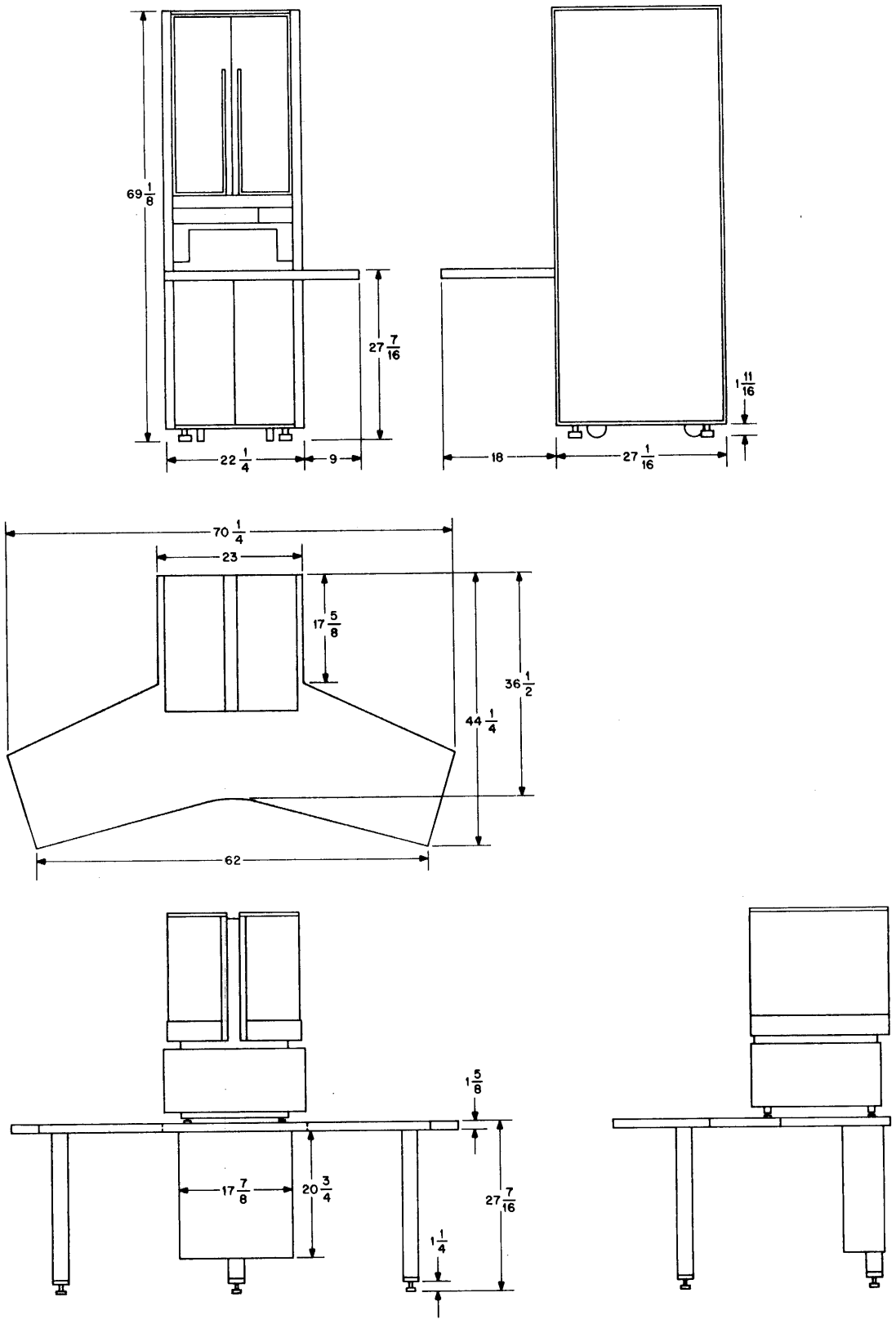



Figure 59 Optional Cabinet and Table Installation Dimensions

The standard Teletype automatic send receive set requires floor space approximately 22-1/4 inches wide by 18-1/2 inches deep. Signal cable length restricts the location of the Teletype to within 18 inches of the side of the computer.

## Environmental Requirements

Ambient temperature at the installation site can vary between 32 and 130 F (between 0 and 55 C) with no adverse effect on computer operation. However, to extend the life expectancy of the system, it is recommended that the ambient temperature at the installation site be maintained between 70 and 85 F (between 21 and 30 C).

During shipping or storing of the system, the ambient temperature may vary between 32 and 130 F (between 0 and 55 C). Although all exposed surfaces of all DEC cabinets and hardware are treated to prevent corrosion, exposure of systems to extreme humidity for long periods of time should be avoided.

## Power Requirements

A source of 115v (±17v), 60-cps (±0.5 cps), single-phase power capable of supplying at least 15 amp must be provided to operate a standard PDP-8. To allow connection to the power cable of the computer, this source should be provided with a Hubbell 3-terminal, except for the basic table top PDP-8 grounded-neutral flush receptacle (or its equivalent). A table-mounted PDP-8 is provided with a 15-amp power plug; a rack-mounted PDP-8 has a 20 amp twist-lock plug; and systems that draw more than 20 amps use a 30-amp twist-lock plug.

Power dissipation of a standard PDP-8 is approximately 780w, and the heat dissipation is approximately 2370 Btu/hr. Upon special request, a PDP-8 can be constructed to operate from a 220v (±33v), 60-cps (±0.5 cps), single-phase power source or from a 100v (±15v), 50-cps (±0.5 cps), single-phase power source.

## Cable Requirements

Nine-conductor coaxial cables with Type W011 Cable Connectors provide signal connection between the computer and optional equipment in free-standing cabinets. These cables are connected by plugging the W011 connectors into standard FLIP CHIP module receptacles.

All free standing cabinets require independent 115v receptacles. However, these units may be turned on or off or controlled from the PDP-8 operator console.

Cables connect to cabinets through a drop panel in the bottom of cabinets. Subflooring is not necessary because casters elevate the cabinets from the floor to afford sufficient cable clearance.

## Installation Procedure

During system check-out, customers are invited to visit the Maynard manufacturing facility to inspect and become familiar with their equipment. Computer customers may also send personnel to instruction courses on computer operation, programming, and maintenance conducted regularly in Maynard, Massachusetts.

DEC's engineers are available during installation and test for assistance or consultation. Further technical assistance in the field is provided by home office design engineers or branch office application engineers.

Table 11 gices installation data to be considered when installing a PDP-8. Table 12 lists space requirements. Figure 60, a typical PDP-8 system configuration, can be used as a guide for planning layout.

TABLE 11 INSTALLATION DATA

| | Weight (lbs) | Dimensions | | | Service Clearance | | Heat Dissipation Btu/hr | Current (amps) | | Power Dissipation (kw) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Height | Width | Depth | Front | Rear | | Nom | Surge | |
| PDP-8 Table Top | 250 | 32 | 21-1/2 | 21-1/4 | - | - | 2660 | 7.5 | - | 0.78 |
| PDP-8 ③ Rack Mount | 250 | 31-1/4 | 19-5/8 ① | 21-7/8 ② | 22 | - | 2660 | 7.5 | - | 0.78 |
| Standard Cabinet CAB8B (Empty) | 225 | 69-1/8 | 22-1/4 | 27-1/16 | 22 | 15 | - | - | - | - |
| Teletype ASR-33 | 40 | 45 | 23 | 19 | - | - | (Included in Standard PDP-8) | | | |
| Serial Drum 251 | 500 | 70 | 23 | 28 | 9 | 15 | 1540 | 5 | 8 | 0.45 |
| Card Reader CR01C | 25 | 8-1/4 | 18 | 10 | - | - | 270 | 0.57 | 1 | 0.06 |
| Card Reader 451A | 225 | 42 | 30 | 18 | - | 7 | 450 | 1.3 | 7 | 0.2 |
| Magnetic Tape Transport 50 | 600 | 69-1/8 | 22-1/4 | 27-1/16 | 18-5/8 | 15 | 2114 | 8 | 12 | 0.96 |
| Magnetic Tape Transport 570 | 850 | 68 | 32-1/8 | 32-3/8 | 19 | 17 | 9900 | 25 | 40 | 2.9 |
| Magnetic Tape Transport 545 | 400 | 69-1/8 | 22-1/4 | 27-1/16 | 9 | 15 | 2870 | 7.3 | 8 | 0.9 |
| Magnetic Tape System 580 | 400 | 69-1/8 | 22-1/4 | 27-1/16 | 9 | 15 | 2870 | 7.3 | 8 | 0.9 |
| DECtape Transport TU55 | 35 | 10-1/2 | 19-1/2 | 9-3/4 | 9 | - | 410 | 1 | 2 | 0.11 |
| Precision Display 30N | 350 | 49 | 53 | 39 | - | 15 | 3140 | 8 | 8 | 0.9 |

NOTES:

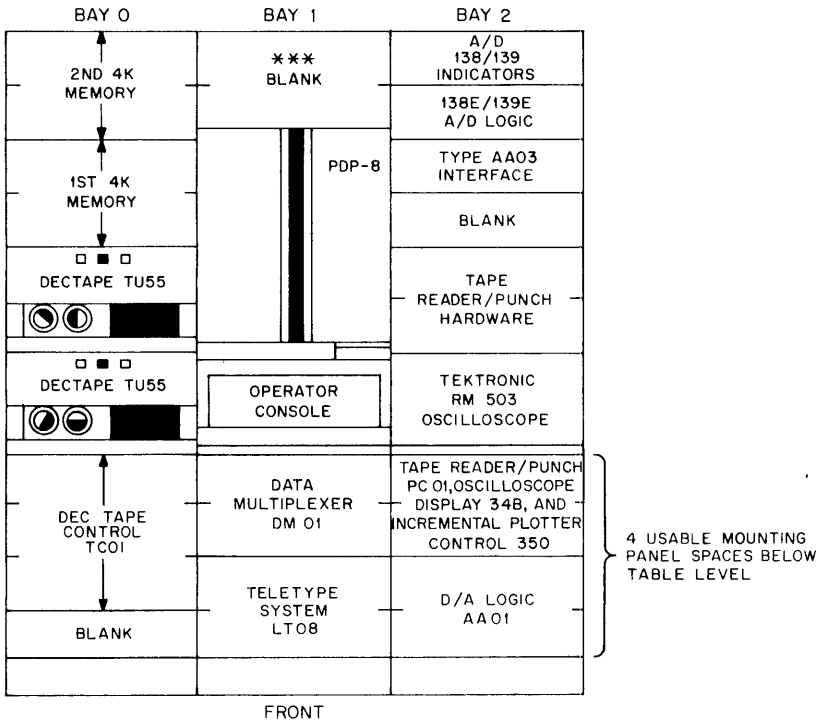
① 19 inch console panel available on request
② Overall depth is 24-3/8 inches from front of console to back of chassis track slices.
③ When PDP-8 is mounted in CAB8 (A or B), there is additional room for:
   a. One standard (5-1/4 inch high) DEC logic mounting panel above the computer in front.
   b. Three standard DEC logic mounting panels below the computer table level in front.
   c. On the rear plenum door, space is available for 1 short mounting panel at the top and three short paneis below table level.

## TABLE 12 SPACE REQUIREMENTS

| Option | MOUNTING PANELS* | | Remarks |
|---|---|---|---|
| | Logic | Other | |
| Memory Extension Control 183 | - | - | Mounts within standard PDP-8 package |
| Memory Module 184 | 2 | - | Should be mounted in expander cabinet next to PDP-8 |
| Automatic Multiply-Divide 182 | - | - | Mounts within standard PDP-8 package |
| Memory Parity 188 | - | - | Mounts within standard PDP-8 package |
| Automatic Restart KR01 | - | - | Mounts within standard PDP-8 package |
| Data Channel Multiplexer DM01 | - | 2 | |
| Perforated Tape Reader 750C | 2 | 10 in. front panel | These 5 options share same two mounting panels for control logic. |
| Perforated Tape Punch 75E | 2 | 15-3/4 in. vertical clearance in cabinet | |
| Oscilloscope Display 34 D | 2 | 10 in. front panel | |
| Incremental Plotter 350B | 2 | Table space needed for plotter | |
| Card Reader CR01C | 2 | Table space needed for reader | |
| A/D Converter 189 | - | - | Mounts within standard PDP-8 package |
| A/D Converter and Multiplexer 138E/139E | 2 | 5-1/4 in. for indicator panel | No additional space needed for 64 channel multiplexer expansion |
| Light Pen 370 | - | - | Logic and power supply included in 34D or 30N |
| DCS 680<br>681<br>685<br>682 | -<br>-<br>2<br>2 | -<br>-<br>-<br>- | Mounted within standard PDP-8 package for up to 64 full duplex channels connectors for up to 64 Teletypes |
| Teletype System LT08 | 2 | - | Handles up to 5 Teletypes |
| Magnetic Tape Control 57A | 7 | 5-1/4 in. connector panel | Controls up to 8 IBM-compatible tape units (570, 50, or 545) |
| DECtape Control TC01 | 3 | - | Controls up to 8 TU55 DECtape Transports |

*Mounting Panels are standard DEC 5-1/4 in. high logic panels.

NOTE: Power supplies for option logic are normally mounted on rear door of DEC cabinets. Customers using their own cabinets should allow additional space for power supplies.



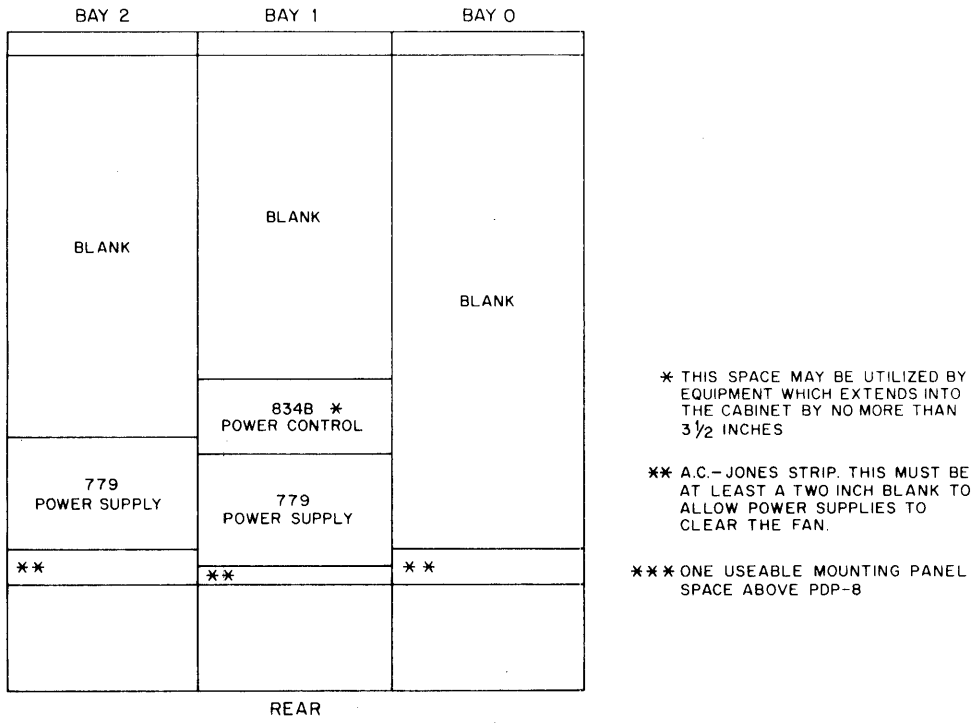




Figure 60 Typical PDP-8 System Configuration and Layoff Planning

# APPENDIX 1
# INSTRUCTIONS

## MEMORY REFERENCE INSTRUCTIONS

| Mnemonic Symbol | Operation Code | Direct Addr. | | Indirect Addr. | | Operation |
|---|---|---|---|---|---|---|
| | | States Entered | Execution Time ($\mu$sec) | States Entered | Execution Time ($\mu$sec) | |
| AND Y | 0 | F, E | 3.0 | F, D, E | 4.5 | Logical AND. The AND operation is performed between the content of memory location Y and the content of the AC. The result is left in the AC, the original content of the AC is lost, and the content of Y is restored. Corresponding bits of the AC and Y are operated upon independently. ACj $\wedge$ Yj $=>$ ACj |
| TAD Y | 1 | F, E | 3.0 | F, D, E | 4.5 | Two's complement add. The content of memory location Y is added to the content of the AC in two's complement arithmetic. The result of this addition is held in the AC, the original content of the AC is lost, and the content of Y is restored. If there is a carry from AC0, the link is complemented. AC + Y $=>$ AC |
| ISZ Y | 2 | F, E | 3.0 | F, D, E | 4.5 | Increment and skip if zero. The content of memory location Y is incremented by one. If the resultant content of Y equals zero, the content of the PC is incremented and the next instruction is skipped. If the resultant content of Y does not equal zero, the program proceeds to the next instruction. The incremented content of Y is restored to memory. If resultant Y = 0, PC + 1 $=>$ PC |

## MEMORY REFERENCE INSTRUCTIONS (continued)

| Mnemonic Symbol | Operation Code | Direct Addr. | | Indirect Addr. | | Operation |
|---|---|---|---|---|---|---|
| | | States Entered | Execution Time ($\mu$sec) | States Entered | Execution Time ($\mu$sec) | |
| DCA Y | 3 | F, E | 3.0 | F, D, E | 4.5 | Deposit and clear AC. The content of the AC is deposited in core memory at address Y and the AC is cleared. The previous content of memory location Y is lost.<br>AC => Y<br>0 => AC |
| JMS Y | 4 | F, E | 3.0 | F, D, E | 4.5 | Jump to subroutine. The content of the PC is deposited in core memory location Y and the next instruction is taken from core memory location Y + 1.<br>PC + 1 => Y<br>Y + 1 => PC |
| JMP Y | 5 | F | 1.5 | F, D | 3.0 | Jump to Y. Address Y is set into the PC so that the next instruction is taken from core memory address Y. The original content of the PC is lost.<br>Y => PC |

# BASIC IOT MICROINSTRUCTIONS

| Mnemonic | Octal | Operation |
|---|---|---|
| | | PROGRAM INTERRUPT |
| ION | 6001 | Turn interrupt on and enable the computer to respond to an interrupt request. When this instruction is given, the computer executes the next instruction, then enables the interrupt. The additional instruction allows exit from the interrupt subroutine before allowing another interrupt to occur. |
| IOF | 6002 | Turn interrupt off i.e. disable the interrupt. |
| | | ANALOG-TO-DIGITAL CONVERTER TYPE 189 |
| ADC | 6004 | Convert the analog input signal to a digital value. |
| | | HIGH SPEED PERFORATED TAPE READER AND CONTROL TYPE 750C |
| RSF | 6011 | Skip if reader flag is a 1. |
| RRB | 6012 | Read the content of the reader buffer and clear the reader flag. (This instruction does not clear the AC.) RB V AC 4-11 => AC 4-11 |
| RFC | 6014 | Clear reader flag and reader buffer, fetch one character from tape and load it into the reader buffer, and set the reader flag when done. |
| | | HIGH SPEED PERFORATED TAPE PUNCH AND CONTROL TYPE 75E |
| PSF | 6021 | Skip if punch flag is a 1. |
| PCF | 6022 | Clear punch flag and punch buffer. |
| PPC | 6024 | Load the punch buffer from bits 4 through 11 of the AC and punch the character. This instruction does not clear the punch flag or punch buffer.) AC 4-11 V PB => PB |
| PLS | 6026 | Clear the punch flag, clear the punch buffer, load the punch buffer from the content of bits 4 through 11 of the accumulator, punch the character, and set the punch flag to 1 when done. |
| | | TELETYPE KEYBOARD/READER |
| KSF | 6031 | Skip if keyboard flag is a 1. |
| KCC | 6032 | Clear AC and clear keyboard flag. |
| KRS | 6034 | Read keyboard buffer static. (This is a static comman in that neither the AC nor the keyboard flag is cleared.) TTI V AC 4-11 => AC 4 -11 |
| KRB | 6036 | Clear AC, clear keyboard flag, and read the content of the keyboard buffer into the content of AC 4-11. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | TELETYPE TELEPRINTER/PUNCH |
| TSF | 6041 | Skip if teleprinter flag is a 1. |
| TCF | 6042 | Clear teleprinter flag. |
| TPC | 6044 | Load the TTO from the content of AC 4-11 and print and/or punch the character. |
| TLS | 6046 | Load the TTO from the content of AC 4-11, clear the teleprinter flag, and print and /or punch the character. |
| | | OSCILLOSCOPE DISPLAY TYPE 34D AND PRECISION CRT DISPLAY TYPE 30N |
| DCX | 6051 | Clear X coordinate buffer. |
| DXL | 6053 | Clear and load X coordinate buffer. AC 2-11 => XB |
| DIX | 6054 | Intensify the point defined by the content of the X and Y coordinate buffers. |
| DXS | 6057 | Executes the combined functions of DXL followed by DIX. |
| DCY | 6061 | Clear Y coordinate buffer. |
| DYL | 6063 | Clear and load Y coordinate buffer. AC 2-11 => YB |
| DIY | 6064 | Intensify the point defined by the content of the X and Y coordinate buffers. |
| DYS | 6067 | Executes the combined functions of DYL followed by DIY. |
| | | OSCILLOSCOPE DISPLAY TYPE 34D |
| DSB | 6075 | Set minimum brightness. |
| DSB | 6076 | Set medium brightness. |
| DSB | 6077 | Set maximum brightness. |
| | | PRECISION CRT DISPLAY TYPE 30N |
| DLB | 6074 | Load brightness register (BR) from bits 9 through 11 of the AC. AC 9-11 => BR |
| | | LIGHT PEN TYPE 370 |
| DSF | 6071 | Skip if display flag is a 1. |
| DCF | 6072 | Clear the display flag. |
| | | MEMORY PARITY TYPE 188 |
| SMP | 6101 | Skip if memory parity error flag = 0. |
| CMP | 6104 | Clear memory parity error flag. |
| | | AUTOMATIC RESTART TYPE KR01 |
| SPL | 6102 | Skip if power is low. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | MEMORY EXTENSION CONTROL TYPE 183 |
| CDF | 62N1 | Change to data field N. The data field register is loaded with the selected field number (0 to 7). All subsequent memory requests for operands are automatically switched to that data field until the data field number is changed by a new CDF command. |
| CIF | 62N2 | Prepare to change to instruction field N. The instruction buffer register is loaded with the selected field number (0 to 7). The next JMP or JMS instruction causes the new field to be entered. |
| RDF | 6214 | Read data field into AC 6-8. Bits 0-5 and 9-11 of the AC are not affected. |
| RIF | 6224 | Same as RDF except reads the instruction field. |
| RIB | 6234 | Read interrupt buffer. The instruction field and data field stored during an interrupt are read into AC 6-8 and 9-11 respectively. |
| RMF | 6244 | Restore memory field. Used to exit from a program interrupt. |
| | | DATA COMMUNICATION SYSTEMS TYPE 630 |
| TTINCR | 6401 | The content of the line select register is incremented by one. |
| TTI | 6402 | The line status word is read and sampled. If the line is active for the fourth time, the line bit is shifted into the character assembly word. If the line is active for a number of times less than four, the count is incremented. If the line is not active, the active/inactive status of the line is recorded. |
| TTO | 6404 | The character in the AC is shifted right one position, zeros are shifted into vacated positions, and the original content of AC11 is transferred out of the computer on the Teletype line. |
| TTCL | 6411 | The line select register is cleared. |
| TTSL | 6412 | The line select register is loaded by an OR transfer from the content of AC5-11, then the AC is cleared. |
| TTRL | 6414 | The content of the line select register is read into AC5-11 by an OR transfer. |
| TTSKP | 6421 | Skip if clock 1 flag is a 1. |
| TTXON | 6422 | Clock 1 is enabled to request a program interrupt and clock 1 flag is cleared. |
| TTXOF | 6424 | Clock 1 is disabled from causing a program interrupt and clock 1 flag is cleared. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | **INCREMENTAL PLOTTER AND CONTROL TYPE 350B** |
| PLSF | 6501 | Skip if plotter flag is a 1. |
| PLCF | 6502 | Clear plotter flag. |
| PLPU | 6504 | Plotter pen up. Raise pen off of paper. |
| PLPR | 6511 | Plotter pen right. |
| PLDU | 6512 | Plotter drum (paper) upward. |
| PLDD | 6514 | Plotter drum (paper) downward. |
| PLPL | 6521 | Plotter pen left. |
| PLUD | 6522 | Plotter drum (paper) upward. (Same as 6512.) |
| PLPD | 6524 | Plotter pen down. Lower pen on to paper. |
| | | **GENERAL PURPOSE CONVERTER TYPE 138E AND MULTIPLEXER CONTROL TYPE 139E** |
| ADSF | 6531 | Skip if A/D converter flag is a 1. |
| ADCV | 6532 | Clear A/D converter flag and convert input voltage to a digital number, flag will set to 1 at end of conversion. Number of bits in converted number determined by switch setting, 11 bits maximum. |
| ADRB | 6534 | Read A/D converter buffer into AC, left justified, and clear flag. |
| ADCC | 6541 | Clear multiplexer channel address register. |
| ADSC | 6542 | Set up multiplexer channel as per AC 6-11. Maximum of 64 single ended or 32 differential input channels. |
| ADIC | 6544 | Index multiplexer channel address (present address + 1). Upon reaching address limit, increment will cause channel 00 to be selected. |
| | | **SERIAL MAGNETIC DRUM SYSTEM TYPE 251** |
| DRCR | 6603 | Load the drum core location counter with the core memory location information in the accumulator. Prepare to read one sector of information from the drum into the specified core location. Then clear the AC. |
| DRCW | 6605 | Load the drum core location counter with the core memory location information in the accumulator. Prepare to write one sector of information into the drum from the specified core location. Then clear the AC. |
| DRCF | 6611 | Clear completion flag and error flag. |
| DREF | 6612 | Clear the AC then load the condition of the parity error and data timing error flip-flops of the drum control into accumulator bits 0 and 1 respectively to allow programmed evaluation of an error flag. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | SERIAL MAGNETIC DRUM SYSTEM TYPE 251 (continued) |
| DRTS | 6615 | Load the drum address register with the track and sector address held in the accumulator. Clear the completion and error flags, and begin a transfer (reading or writing). Then clear the AC. |
| DRSE | 6621 | Skip next instruction if the error flag is a 0 (no error). |
| DRSC | 6622 | Skip next instruction if the completion flag is a 1 (sector transfer is complete). |
| DRCN | 6624 | Clear error flag and completion flag, then initiate transfer of next sector. |
| | | CARD READER AND CONTROL TYPE CRO1C |
| RCSF | 6631 | Skip if card reader data ready flag is a 1. |
| RCRA | 6632 | The alphanumeric code for the column is read into AC6-11, and the data ready flag is cleared. |
| RCRB | 6634 | The binary data in a card column is transferred into AC0-11, and the data ready flag is cleared. |
| RCSP | 6671 | Skip if card reader card done flag is a 1. |
| RCSE | 6672 | Clear the card done flag, select the card reader and start card motion towards the read station, and skip if the reader-not-ready flag is a 1. |
| RCRD | 6674 | Clear card done flag. |
| | | CARD READER AND CONTROL TYPE 451 |
| CRSF | 6632 | Skip if card reader flag is a 1. If a card column is present for reading, the next instruction is skipped. |
| CERS | 6634 | Card equipment read status. Reads the status of the card reader flag and status levels into bits 6 through 9 of the AC. The AC bit assignments are:<br>AC6 = Flag is set to 1 (the flag rises after reading each of the 80 rows).<br>AC7 = Card done.<br>AC8 = Not ready (covers not in place, power is off, START button has not been pressed, hopper is empty, stacker is full, a card is jammed, a validity check error has been detected, or the read circuit is defective).<br>AC9 = End of file (EOF) (hopper is empty and operator has pushed START button). |
| CRRB | 6671 | Read the card column buffer information into the AC and clear the card reader flag. One CRRB command reads alphanumeric information. Two CRRB instructions read the upper and lower column binary information. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | CARD READER AND CONTROL TYPE 451 (continued) |
| CRSA | 6672 | Select a card in alphanumeric mode. Select the card reader and start a card moving. Information appears in alphanumeric form. |
| CRSB | 6674 | Select a card in binary mode. Select the card reader and start a card moving. Information appears in binary form. |
| | | CARD PUNCH AND CONTROL TYPE 450 |
| CPSF | 6631 | Skip if card punch flag is a 1. The card punch flag indicates the punch buffer is available, and should be loaded. |
| CERS | 6634 | Card equipment read status. Reads the status of the card punch flag and the card punch error level into bits 10 and 11 of the AC, respectively. |
| CPCF | 6641 | Clear card punch flag. |
| CPSE | 6642 | Select the card punch. Transmit a card to the 80-column punch die from the hopper. |
| CPLB | 6644 | Load the card punch buffer from the content of the AC. Seven load instructions must be given to fill the buffer. |
| | | AUTOMATIC LINE PRINTER AND CONTROL TYPE 645 |
| LSE | 6651 | Skip if line printer error flag is a 1. |
| LCB | 6652 | Clear both sections of the printing buffer. |
| LLB | 6654 | Load printing buffer from the content of AC 6-11 and clear the AC. |
| LSD | 6661 | Skip if the printer done flag is a 1. |
| LCF | 6662 | Clear line printer done and error flags. |
| LPR | 6664 | Clear the format register, load the format register from the content of AC 9-11, print the line contained in the section of the printer buffer loaded last, clear the AC, and advance the paper in accordance with the selected channel of the format tape if the content of AC 8 = 1. If the content of AC 8 = 0, the line is printed and paper advance is inhibited. |
| | | AUTOMATIC MAGNETIC TAPE CONTROL TYPE 57A |
| MSCR | 6701 | Skip if the tape control ready (TCR) level is 1. A 1 is added to the content of the program counter if the tape control is free to accept a command. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | AUTOMATIC MAGNETIC TAPE CONTROL TYPE 57A (continued) |
| MCD | 6702 | Clear the job done flag, clear command register, clear word count overflow (WCO) flag, and clear end of record (EOR) flag. This instruction should be immediately preceded by the two instructions CLA and TAD (4000) to obtain the operation indicated. The job done flag is connected to the program interrupt facility. |
| MTS | 6706 | Disable the job done flag from the program interrupt, turn on the WCO flag and EOR flag and select the unit, the mode of parity, and the density from the content of the AC. The AC bit assignments are:<br><br>(Type 521 and 522 interface only)<br>AC1(0) = High sense level<br>AC1(1) = Low sense level<br><br>AC2(0) = 200 or 556 density<br>AC2(1) = 800 or 556 density<br><br>AC8(0) = 200 density<br>AC8(1) = 556 density |

| AC2 | AC8 | Density |
|---|---|---|
| 0 | 0 | 200 |
| 0 | 1 | 556 |
| 1 | 0 | 800 |
| 1 | 1 | 556 |

AC7(0) = Even parity (BCD)
AC7(1) = Odd parity (binary)

AC9-11 These three bits select one of eight tape units, addresses 0-7.

| | | |
|---|---|---|
| MSUR | 6711 | Skip if the tape transport is ready (TTR). The selected tape unit is checked, using this command, and must be free before the following MTC command is given. |
| MNC | 6712 | Terminate the continuous mode. This instruction clears the AC at completion. It should be immediately preceded by the two instructions CLA and TAD (4000) to obtain the operation indicated. |
| MTC | 6716 | Place the content of AC 3-6 in the tape control command register and start tape motion. Bit 6 selects motion mode.<br><br>AC6(0) = Normal<br>AC6(1) = Continuous |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | AUTOMATIC MAGNETIC TAPE CONTROL TYPE 57A (continued) |
| MTC | 6716 (cont.) | AC3-5 are decoded as follows:<br>0 = No operation<br>1 = Rewind<br>2 = Write<br>3 = Write end of file (EOF)<br>4 = Read compare<br>5 = Read<br>6 = Space forward<br>7 = Space backward |
| MSWF | 6721 | Skip if the WCO flag is a 1. The WCO flag is connected to the program interrupt. |
| MDWF | 6722 | Disable WCO flag. |
| MCWF | 6722 | Clear WCO flag. This instruction should be immediately preceded by the two instructions CLA and TAD (2000) to obtain the operation indicated. |
| MEWF | 6722 | Enable WCO flag. This instruction should be immediately preceded by the two instructions CLA and TAD (4000) to obtain the operation indicated. |
| MIWF | 6722 | Initialize WCO flag. This instruction should be immediately preceded by the two instructions CLA and TAD (6000) to obtain the operation indicated. |
| MSEF | 6731 | Skip if the EOR flag is a 1. This flag is connected to the program interrupt. |
| MDEF | 6732 | Disable ERF. |
| MCED | 6732 | Clear ERF. This instruction should be immediately preceded by the two instructions CLA and TAD (2000) to obtain the operation indicated. |
| MEEF | 6732 | Enable ERF. This instruction should be immediately preceded by the two instructions CLA and TAD (4000) to obtain the operation indicated. |
| MIEF | 6732 | Initialize ERF, clear and enable. This instruction should be immediately preceded by the two instructions CLA and TAD (6000) to obtain the operation indicated. |
| MTRS | 6734 | Read tape status bits into the content of the AC. This instruction should be immediately preceded by a CLA instruction to obtain the operation indicated. The bit assignments are:<br>0 = Data request late<br>1 = Tape parity error<br>2 = Read compare error<br>3 = End of file flag set<br>4 = Write lock ring out<br>5 = Tape at load point |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | AUTOMATIC MAGNETIC TAPE CONTROL TYPE 57A (continued) |
| MTRS | 6734 (cont.) | 6 = Tape at end point<br>7 = Tape near end point (Type 520)<br>7 = Last operation write (Type 521 and 522 interface)<br>8 = Tape near load point (Type 520)<br>8 = Write echo (Type 522 interface)<br>8 = B control using transporting (Type 521 interface with multiplex transport)<br>9 = Transport rewinding<br>10 = Tape miss character<br>11 = Job done flag interrupt |
| MCC | 6741 | Clear CA and WC. |
| MRWC | 6742 | Transfer the content of AC into the WC. |
| MRCA | 6744 | Transfer the content of the CA into the AC. This instruction should be immediately preceded by a CLA command to obtain the operation indicated. |
| MCA | 6745 | Clear CA and WC, and transfer the content of the AC into the CA. |
| | | MAGNETIC TAPE SYSTEM TYPE 580 |
| TIFM | 6707 | Tape initialize function and motion. Clears all tape control registers, loads the command register from the content of the AC, and initiates motion delay. The bit assignments of the command register are:<br><br>AC1 = Space<br>AC3 = Go<br>AC4 = Write<br>AC5 = Parity mode (0 = even, 1 = odd)<br>AC6 = Read<br>AC7 = Direction (0 = reverse, 1 = forward)<br>AC8 = Density (0 = 200 BPI, 1 = 556 BPI)<br>AC10 = Rewind<br>AC11 = Real time |
| TSRD | 6715 | Tape system read. Clear the AC then load the AC from the content of the data buffer and clear the data flag. |
| TSWR | 6716 | Tape system write. Clear the data buffer, then load the data buffer from the content of the AC and clear the data flag. |
| TSDF | 6721 | Tape skip on data flag. If the data flag is a 1, the next instruction is skipped. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | MAGNETIC TAPE SYSTEM TYPE 580 (continued) |
| TSSR | 6722 | Tape skip on end of record. If the end of record delay is a 0 or if the data flag is a 1, the next instruction is skipped. |
| TSST | 6724 | Tape system stop data transfer. This instruction is issued following transmission of the last character in a record. It initiates tape shut down procedures such as writing the longitudinal parity bit, end of record mark, and the 0.75-inch inter-record gap. It also clears the SPACE flip-flop. |
| TWRT | 6731 | Tape system write real time. One character is written on tape. This instruction can be used at any frequency and therefore determines the density of information written on tape. |
| TCPI | 6732 | Tape clear program interrupt. The program interrupt request flag is sampled and if it is a 1 the next instruction is skipped. This command also clears the program interrupt request flag in the control during a space operation and clears the STOP flip-flop. |
| TSRS | 6734 | Tape system read status. The content of the status register is transferred into the AC. The bit assignments are:<br>AC0(1) = Parity error<br>AC1(1) = Motion delay set<br>AC2(1) = Transport is ready<br>AC3(1) = Clock delays set<br>AC4(1) = End of tape<br>AC5(1) = Tape at load point |
| | | DECTAPE DUAL TRANSPORT TYPE 555 AND DECTAPE CONTROL TYPE 552 |
| MMLS | 6751 | Load unit select register from the content of AC 2-5 and set DECtape (DT) flag when done. |
| MMLM | 6752 | Load motion register from the content of AC7-8 and set DT flag when done. |
| MMLF | 6754 | Load function register from the content of AC 9-11 then clear the AC. The octal code of these three bits establishes the following DECtape control modes:<br>0 = Move<br>1 = Search<br>2 = Read data<br>3 = Read all bits<br>4 = Write data<br>5 = Write all bits<br>6 = Write mark and timing |
| MMMF | 6756 | Load motion register from AC7-8, load function register from AC9-11, clear the AC, and set the DT flag when done. |

| Mnemonic | Octal | Operation |
|---|---|---|
| | | **DECTAPE DUAL TRANSPORT TYPE 555 AND DECTAPE CONTROL TYPE 552 (continued)** |
| MMMM | 6757 | Load the unit select register, motion register, and function register from AC2-11; clear the AC, and set the DT flag when done. |
| MMSF | 6761 | Skip if DT flag is a 1. |
| MMCC | 6762 | Clear memory address counter (MAC). |
| MMLC | 6764 | Load MAC from the content of ACO-11 and then clear the AC. |
| MMML | 6766 | Clear MAC, load MAC from ACO-11, and clear AC. |
| MMSC | 6771 | Skip if error flag is a 1. |
| MMCF | 6772 | Clear error flag and DT flag. |
| MMRS | 6774 | Read status bits into the content of AC 0-7. The bit assignments are:<br>AC0 = DT flag<br>AC1 = Error flag<br>AC2 = End (selected tape at end point)<br>AC3 = Timing error<br>AC4 = Reverse tape direction<br>AC5 = Go<br>AC6 = Parity or mark track error<br>AC7 = Select error |
| | | **DECTAPE TRANSPORT TYPE TU55 AND DECTAPE CONTROL TYPE TC01** |
| DTRA | 6761 | The content of status register A is read into AC0-9 by an OR transfer. The bit assignments are:<br>AC0-2 = Transport unit select number<br>AC3-4 = Motion<br>AC5 = Mode<br>AC6-8 = Function<br>AC9 = Enable/disable DECtape control flag |
| DTCA | 6762 | Clear status register A. All flags undisturbed. |
| DTXA | 6764 | Status register A is loaded by an exclusive OR transfer from the content of the AC, and AC10 and AC11 are sampled. If AC10 = 0, the error flags are cleared. If AC11 = 0, the DECtape control flag is cleared. |
| DTSF | 6771 | Skip if error flag is a 1 or if DECtape control flag is a 1. |

## BASIC IOT MICROINSTRUCTIONS (continued)

| Mnemonic | Octal | Operation |
|---|---|---|
| | | DECTAPE TRANSPORT TYPE TU55 AND<br>DECTAPE CONTROL TYPE TC01 (continued) |
| DTRB | 6772 | The content of status register B is read into the AC by an OR transfer. The bit assignments are:<br>AC0 = Error flag<br>AC1 = Mark track error<br>AC2 = End of tape<br>AC3 = Select error<br>AC4 = Parity error<br>AC5 = Timing error<br>AC6-8 = Memory field<br>AC9-10 = Unused<br>AC11 = DECtape flag |
| DTLB | 6774 | The memory field portion of status register B is loaded by an OR transfer from the content of AC6-8. |

# GROUP 1 OPERATE MICROINSTRUCTIONS

| Mnemonic Symbol | Octal Code | Event Time | Operation |
|---|---|---|---|
| NOP | 7000 | — | No operation. Causes a 1.5 $\mu$sec program delay. |
| IAC | 7001 | 2 | Increment AC. The content of the AC is incremented by one in two's complement arithmetic. |
| RAL | 7004 | 2 | Rotate AC and L left. The content of the AC and the L are rotated left one place. |
| RTL | 7006 | 2 | Rotate two places to the left. Equivalent to two successive RAL operations. |
| RAR | 7010 | 2 | Rotate AC and L right. The content of the AC and L are rotated right one place. |
| RTR | 7012 | 2 | Rotate two places to the right. Equivalent to two successive RAR operations. |
| CML | 7020 | 1 | Complement L. |
| CMA | 7040 | 1 | Complement AC. The content of the AC is set to the one's complement of its current content. |
| CIA | 7041 | 1, 2 | Complement and increment accumulator. Used to form two's complement. |
| CLL | 7100 | 1 | Clear L. |
| CLL RAL | 7104 | 1, 2 | Shift positive number one left. |
| CLL RTL | 7106 | 1, 2 | Clear link, rotate two left. |
| CLL RAR | 7110 | 1, 2 | Shift positive number one right. |
| CLL RTR | 7112 | 1, 2 | Clear link, rotate two right. |
| STL | 7120 | 1 | Set link. The L is set to contain a binary 1. |
| CLA | 7200 | 1 | Clear AC. To be used alone or in OPR 1 combinations. |
| CLA IAC | 7201 | 1, 2 | Set AC = 1 |
| GLK | 7204 | 1, 2 | Get link. Transfer L into AC 11. |
| CLA CLL | 7300 | 1 | Clear AC and L. |
| STA | 7240 | 1 | Set AC = − 1. Each bit of the AC is set to contain a 1. |

# GROUP 2 OPERATE MICROINSTRUCTIONS

| Mnemonic Symbol | Octal Code | Event Time | Operation |
|---|---|---|---|
| HLT | 7402 | 1 | Halt. Stops the program after completion of the cycle in process. If this instruction is combined with others in the OPR 2 group the other operations are completed before the end of the cycle. |
| OSR | 7404 | 2 | OR with switch register. The OR function is performed between the content of the SR and the content of the AC, with the result left in the AC. |
| SKP | 7410 | 1 | Skip, unconditional. The next instruction is skipped. |
| SNL | 7420 | 1 | Skip if L $\neq$ 0. |
| SZL | 7430 | 1 | Skip if L = 0. |
| SZA | 7440 | 1 | Skip if AC = 0. |
| SNA | 7450 | 1 | Skip if AC $\neq$ 0. |
| SZA SNL | 7460 | 1 | Skip if AC = 0, or L = 1, or both. |
| SNA SZL | 7470 | 1 | Skip if AC $\neq$ 0 and L = 0. |
| SMA | 7500 | 1 | Skip on minus AC. If the content of the AC is a negative number, the next instruction is skipped. |
| SPA | 7510 | 1 | Skip on positive AC. If the content of the AC is a positive number, the next instruction is skipped. |
| SMA SNL | 7520 | 1 | Skip if AC $<$ 0, or L = 1, or both. |
| SPA SZL | 7530 | 1 | Skip if AC $\geq$ 0 and if L = 0. |
| SMA SZA | 7540 | 1 | Skip if AC $\leq$ 0. |
| SPA SNA | 7550 | 1 | Skip if AC $>$ 0. |
| CLA | 7600 | 2 | Clear AC. To be used alone or in OPR 2 combinations. |
| LAS | 7604 | 1 | Load AC with SR. |
| SZA CLA | 7640 | 1 | Skip if AC = 0, then clear AC. |
| SNA CLA | 7650 | 1 | Skip if AC $\neq$ 0, then clear AC. |
| SMA CLA | 7700 | 1 | Skip if AC $<$ 0, then clear AC. |
| SPA CLA | 7710 | 1 | Skip if AC $\geq$ 0, then clear AC. |

# EXTENDED ARITHMETIC ELEMENT MICROINSTRUCTIONS

| Mnemonic Symbol | Octal Code | Event Time | Operation |
|---|---|---|---|
| MUY | 7405 | 2 | Multiply. The number held in the MQ is multiplied by the number held in core memory location PC + 1 (or the next successive core memory location after the MUY command). At the conclusion of this command the most significant 12 bits of the product are contained in the AC and the least significant 12 bits of the product are contained in the MQ.<br>Y x MQ => AC, MQ |
| DVI | 7407 | 2 | Divide. The 24-bit dividend held in the AC (most significant 12 bits) and the MQ (least significant 12 bits) is divided by the number held in core memory location PC + 1 (or the next successive core memory location following the DVI command). At the conclusion of this command the quotient is held in the MQ, the remainder is in the AC, and the L contains a 0. If the L contains a 1, divide overflow occurred so the operation was concluded after the first cycle of the division.<br>AC, MQ ÷ Y => MQ |
| NMI | 7411 | 2 | Normalize. This instruction is used as part of the conversion of a binary number to a fraction and an exponent for use in floating-point arithmetic. The combined content of the AC and the MQ is shifted left by this one command until the content of AC0 is not equal to the content of AC1, to form the fraction. Zeros are shifted into vacated MQ11 positions for each shift. At the conclusion of this operation, the step counter contains a number equal to the number of shifts performed. The content of L is lost.<br>ACj => ACj − 1<br>AC0 => L<br>MQ 0 => AC11<br>MQj = MQj −1<br>0 => MQ11 until AC0 ≠ AC1 |
| SHL | 7413 | 2 | Shift arithmetic left. This instruction shifts the combined content of the AC and MQ to the left one position more than the number of positions indicated by the content of core memory at address PC + 1 (or the next successive core memory location following the SHL command). During the shifting, zeros are shifted into vacated MQ11 positions.<br>Shift Y + 1 positions as follows:<br>ACj => ACj −1<br>AC0 => L<br>MQ0 => AC11<br>MQj => MQj −1<br>0 => MQ11 |

## EXTENDED ARITHMETIC ELEMENT MICROINSTRUCTIONS
(continued)

| Mnemonic Symbol | Octal Code | Event Time | Operation |
|---|---|---|---|
| ASR | 7415 | 2 | Arithmetic shift right. The combined content of the AC and the MQ is shifted right one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the ASR command). The sign bit, contained in AC0, enters vacated positions, the sign bit is preserved, information shifted out of MQ11 is lost, and the L is undisturbed during this operation.<br>Shift Y + 1 positions as follows:<br>AC0 => AC0<br>ACj => ACj + 1<br>AC11 => MQ0<br>MQj => MQj + 1 |
| LSR | 7417 | 2 | Logical shift right. The combined content of the AC and MQ is shifted left one position more than the number contained in memory location PC + 1 (or the next successive core memory location following the LSR command). This command is similar to the ASR command except that zeros enter vacated positions instead of the sign bit entering these locations. Information shifted out of MQ11 is lost and the L is undisturbed during this operation.<br>Shift Y + 1 positions as follows:<br>0 => AC0<br>ACj => ACj + 1<br>AC11 => MQ0<br>MQj => MQj + 1 |
| MQL | 7421 | 2 | Load multiplier quotient. This command clears the MQ, loads the content of the AC into the MQ, then clears the AC.<br>0 => MQ<br>AC => MQ<br>0 => AC |
| SCA | 7441 | 2 | Step counter load into accumulator. The content of the step counter is transferred into the AC. The AC should be cleared prior to issuing this command or the CLA command can be combined with the SCA to clear the AC, then effect the transfer.<br>SC V AC => AC |
| MQA | 7501 | 2 | Multiplier quotient load into accumulator. The content of the MQ is transferred into the AC. This command is given to load the 12 least significant bits of the product into the AC following a multiplication or to load the quotient into the AC following a division. The AC should be cleared prior to issuing this command or the CLA command can be combined with the MQA to clear the AC then effect the transfer.<br>MQ V AC => AC |

## EXTENDED ARITHMETIC ELEMENT MICROINSTRUCTIONS
(continued)

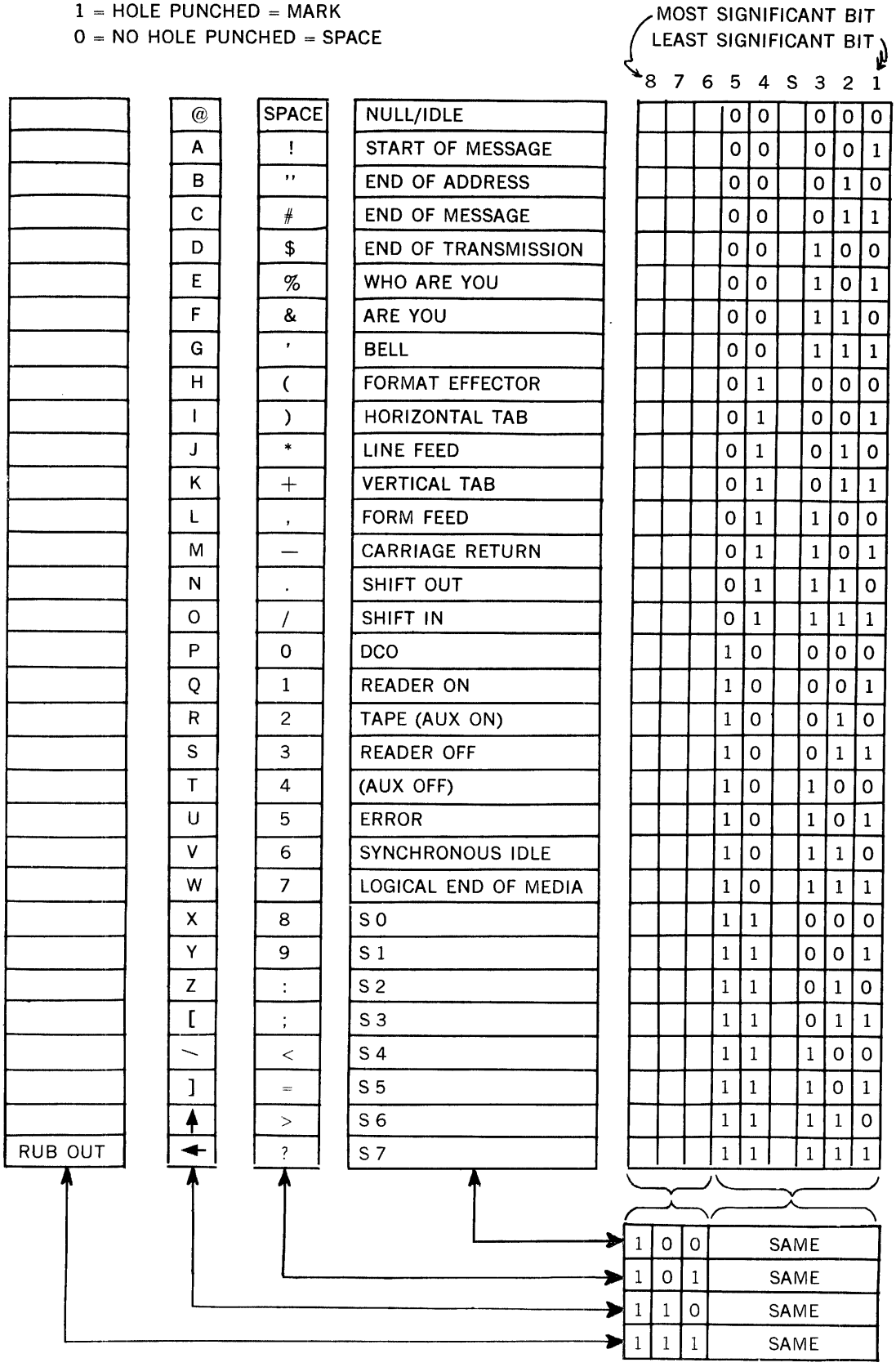
| Mnemonic Symbol | Octal Code | Event Time | Operation |
|---|---|---|---|
| CLA | 7601 | 1 | Clear accumulator. The AC is cleared during event time 1, allowing this command to be combined with the other EAE commands that load the AC during event time 2 (such as SCA and MQA).<br>0 => AC |
| CAM | 7621 | 1, 2 | Clear accumulator and multiplier quotient,<br>CAM = CLA LMQ. |

# APPENDIX 2
# CODES
## MODEL 33 ASR/KSR TELETYPE CODE (ASCII) IN OCTAL FORM

| Character | 8-Bit Code (in octal) | Character | 8-Bit Code (in octal) |
|---|---|---|---|
| A | 301 | ! | 241 |
| B | 302 | " | 242 |
| C | 303 | # | 243 |
| D | 304 | $ | 244 |
| E | 305 | % | 245 |
| F | 306 | & | 246 |
| G | 307 | ' | 247 |
| H | 310 | ( | 250 |
| I | 311 | ) | 251 |
| J | 312 | * | 252 |
| K | 313 | + | 253 |
| L | 314 | , | 254 |
| M | 315 | – | 255 |
| N | 316 | . | 256 |
| O | 317 | / | 257 |
| P | 320 | : | 272 |
| Q | 321 | ; | 273 |
| R | 322 | < | 274 |
| S | 323 | = | 275 |
| T | 324 | > | 276 |
| U | 325 | ? | 277 |
| V | 326 | @ | 300 |
| W | 327 | [ | 333 |
| X | 330 | \ | 334 |
| Y | 331 | ] | 335 |
| Z | 332 | ↑ | 336 |
| | | ← | 337 |
| 0 | 260 | | |
| 1 | 261 | Leader/Trailer | 200 |
| 2 | 262 | Line-Feed | 212 |
| 3 | 263 | Carriage-Return | 215 |
| 4 | 264 | Space | 240 |
| 5 | 265 | Rub-out | 377 |
| 6 | 266 | Blank | 000 |
| 7 | 267 | | |
| 8 | 270 | | |
| 9 | 271 | | |

# MODEL 33 ASR/KSR TELETYPE CODE (ASCII) IN BINARY FORM

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




| | | | | 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 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | SAME |
| 1 | 0 | 1 | SAME |
| 1 | 1 | 0 | SAME |
| 1 | 1 | 1 | SAME |

# CARD READER CODE

| Card Code | | Internal Code | Character |
|---|---|---|---|
| Zone | Num. | | |
| — | — | 01 0000 | Blank |
| 12 | 8-3 | 11 1011 | . |
| 12 | 8-4 | 11 1100 | ) |
| 12 | 8-5 | 11 1101 | ] |
| 12 | 8-6 | 11 1110 | < |
| 12 | 8-7 | 11 1111 | ◄ |
| 12 | — | 11 0000 | + |
| 11 | 8-3 | 10 1011 | $ |
| 11 | 8-4 | 10 1100 | * |
| 11 | 8-5 | 10 1101 | [ |
| 11 | 8-6 | 10 1110 | > |
| 11 | 8-7 | 10 1111 | & |
| 11 | — | 10 0000 | — |
| 0 | 1 | 01 0001 | / |
| 0 | 8-3 | 01 1011 | , |
| 0 | 8-4 | 01 1100 | ( |
| 0 | 8-5 | 01 1101 | " |
| 0 | 8-6 | 01 1110 | # |
| 0 | 8-7 | 01 1111 | % |
| — | 8-3 | 00 1011 | = |
| — | 8-4 | 00 1100 | @ |
| — | 8-5 | 00 1101 | ▲ |
| — | 8-6 | 00 1110 | ' |
| — | 8-7 | 00 1111 | ≠ |
| 12 | 0 | 11 1010 | ? |
| 12 | 1 | 11 0001 | A |
| 12 | 2 | 11 0010 | B |
| 12 | 3 | 11 0011 | C |
| 12 | 4 | 11 0100 | D |
| 12 | 5 | 11 0101 | E |
| 12 | 6 | 11 0110 | F |
| 12 | 7 | 11 0111 | G |
| 12 | 8 | 11 1000 | H |
| 12 | 9 | 11 1001 | I |

| Card Code | | Internal Code | Character |
|---|---|---|---|
| Zone | Num. | | |
| 11 | 0 | 10 1010 | ▲ |
| 11 | 1 | 10 0001 | J |
| 11 | 2 | 10 0010 | K |
| 11 | 3 | 10 0011 | L |
| 11 | 4 | 10 0100 | M |
| 11 | 5 | 10 0101 | N |
| 11 | 6 | 10 0110 | O |
| 11 | 7 | 10 0111 | P |
| 11 | 8 | 10 1000 | Q |
| 11 | 9 | 10 1001 | R |
| 0 | 8-2 | 01 1010 | ; |
| 0 | 2 | 01 0010 | S |
| 0 | 3 | 01 0011 | T |
| 0 | 4 | 01 0100 | U |
| 0 | 5 | 01 0101 | V |
| 0 | 6 | 01 0110 | W |
| 0 | 7 | 01 0111 | X |
| 0 | 8 | 01 1000 | Y |
| 0 | 9 | 01 1001 | Z |
| — | 0 | 00 1010 | 0 |
| — | 1 | 00 0001 | 1 |
| — | 2 | 00 0010 | 2 |
| — | 3 | 00 0011 | 3 |
| — | 4 | 00 0100 | 4 |
| — | 5 | 00 0101 | 5 |
| — | 6 | 00 0110 | 6 |
| — | 7 | 00 0111 | 7 |
| — | 8 | 00 1000 | 8 |
| — | 9 | 00 1001 | 9 |
| All other codes | | 00 0000 | ◄ |

# AUTOMATIC LINE PRINTER CODE

| Character (ASCII) | 6-Bit Code (in octal) | Character (ASCII) | 6-Bit Code (in octal) |
|---|---|---|---|
| @ | 0 | □ | 40 |
| A | 1 | ! | 41 |
| B | 2 | " | 42 |
| C | 3 | # | 43 |
| D | 4 | $ | 44 |
| E | 5 | % | 45 |
| F | 6 | & | 46 |
| G | 7 | ' | 47 |
| H | 10 | ( | 50 |
| I | 11 | ) | 51 |
| J | 12 | * | 52 |
| K | 13 | + | 53 |
| L | 14 | , | 54 |
| M | 15 | − | 55 |
| N | 16 | . | 56 |
| O | 17 | / | 57 |
| P | 20 | ∅ | 60 |
| Q | 21 | 1 | 61 |
| R | 22 | 2 | 62 |
| S | 23 | 3 | 63 |
| T | 24 | 4 | 64 |
| U | 25 | 5 | 65 |
| V | 26 | 6 | 66 |
| W | 27 | 7 | 67 |
| X | 30 | 8 | 70 |
| Y | 31 | 9 | 71 |
| Z | 32 | : | 72 |
| [ | 33 | ; | 73 |
| \ | 34 | < | 74 |
| ] | 35 | = | 75 |
| ↑ | 36 | > | 76 |
| ← | 37 | ? | 77 |

# APPENDIX 3

# SCALES OF NOTATION

## $2^x$ IN DECIMAL

| x | $2^x$ | x | $2^x$ | x | $2^x$ |
|---|---|---|---|---|---|
| 0.001 | 1.00069 33874 62581 | 0.01 | 1.00695 55500 56719 | 0.1 | 1.07177 34625 36293 |
| 0.002 | 1.00138 72557 11335 | 0.02 | 1.01395 94797 90029 | 0.2 | 1.14869 83549 97035 |
| 0.003 | 1.00208 16050 79633 | 0.03 | 1.02101 21257 07193 | 0.3 | 1.23114 44133 44916 |
| 0.004 | 1.00277 64359 01078 | 0.04 | 1.02811 38266 56067 | 0.4 | 1.31950 79107 72894 |
| 0.005 | 1.00347 17485 09503 | 0.05 | 1.03526 49238 41377 | 0.5 | 1.41421 35623 73095 |
| 0.006 | 1.00416 75432 38973 | 0.06 | 1.04246 57608 41121 | 0.6 | 1.51571 65665 10398 |
| 0.007 | 1.00486 38204 23785 | 0.07 | 1.04971 66836 23067 | 0.7 | 1.62450 47927 12471 |
| 0.008 | 1.00556 05803 98468 | 0.08 | 1.05701 80405 61380 | 0.8 | 1.74110 11265 92248 |
| 0.009 | 1.00625 78234 97782 | 0.09 | 1.06437 01824 53360 | 0.9 | 1.86606 59830 73615 |

## $10^{\pm n}$ IN OCTAL

| $10^n$ | n | $10^{-n}$ | $10^n$ | n | $10^{-n}$ |
|---|---|---|---|---|---|
| 1 | 0 | 1.000 000 000 000 000 000 000 00 | 112 402 762 000 | 10 | 0.000 000 000 006 676 337 66 |
| 12 | 1 | 0.063 146 314 631 463 146 31 | 1 351 035 564 000 | 11 | 0.000 000 000 000 537 657 77 |
| 144 | 2 | 0.005 075 341 217 270 243 66 | 16 432 451 210 000 | 12 | 0.000 000 000 000 043 136 32 |
| 1 750 | 3 | 0.000 406 111 564 570 651 77 | 221 411 634 520 000 | 13 | 0.000 000 000 000 003 411 35 |
| 23 420 | 4 | 0.000 032 155 613 530 704 15 | 2 657 142 036 440 000 | 14 | 0.000 000 000 000 000 264 11 |
| 303 240 | 5 | 0.000 002 476 132 610 706 64 | 34 327 724 461 500 000 | 15 | 0.000 000 000 000 000 022 01 |
| 3 641 100 | 6 | 0.000 000 206 157 364 055 37 | 434 157 115 760 200 000 | 16 | 0.000 000 000 000 000 001 63 |
| 46 113 200 | 7 | 0.000 000 015 327 745 152 75 | 5 432 127 413 542 400 000 | 17 | 0.000 000 000 000 000 000 14 |
| 575 360 400 | 8 | 0.000 000 001 257 143 561 06 | 67 405 553 164 731 000 000 | 18 | 0.000 000 000 000 000 000 01 |
| 7 346 545 000 | 9 | 0.000 000 000 104 560 276 41 | | | |

## $n \log_{10} 2$, $n \log_2 10$ IN DECIMAL

| n | $n \log_{10} 2$ | $n \log_2 10$ | n | $n \log_{10} 2$ | $n \log_2 10$ |
|---|---|---|---|---|---|
| 1 | 0.30102 99957 | 3.32192 80949 | 6 | 1.80617 99740 | 19.93156 85693 |
| 2 | 0.60205 99913 | 6.64385 61898 | 7 | 2.10720 99696 | 23.25349 66642 |
| 3 | 0.90308 99870 | 9.96578 42847 | 8 | 2.40823 99653 | 26.57542 47591 |
| 4 | 1.20411 99827 | 13.28771 23795 | 9 | 2.70926 99610 | 29.89735 28540 |
| 5 | 1.50514 99783 | 16.60964 04744 | 10 | 3.01029 99566 | 33.21928 09489 |

## ADDITION AND MULTIPLICATION TABLES

Addition | Multiplication

### Binary Scale

$$\begin{aligned} 0+0&=0 \\ 0+1=1+0&=1 \\ 1+1&=10 \end{aligned}$$

$$\begin{aligned} 0\times 0&=0 \\ 0\times 1=1\times 0&=0 \\ 1\times 1&=1 \end{aligned}$$

### Octal Scale

| 0 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
|---|---|---|---|---|---|---|---|
| 1 | 02 | 03 | 04 | 05 | 06 | 07 | 10 |
| 2 | 03 | 04 | 05 | 06 | 07 | 10 | 11 |
| 3 | 04 | 05 | 06 | 07 | 10 | 11 | 12 |
| 4 | 05 | 06 | 07 | 10 | 11 | 12 | 13 |
| 5 | 06 | 07 | 10 | 11 | 12 | 13 | 14 |
| 6 | 07 | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

| 1 | 02 | 03 | 04 | 05 | 06 | 07 |
|---|---|---|---|---|---|---|
| 2 | 04 | 06 | 10 | 12 | 14 | 16 |
| 3 | 06 | 11 | 14 | 17 | 22 | 25 |
| 4 | 10 | 14 | 20 | 24 | 30 | 34 |
| 5 | 12 | 17 | 24 | 31 | 36 | 43 |
| 6 | 14 | 22 | 30 | 36 | 44 | 52 |
| 7 | 16 | 25 | 34 | 43 | 52 | 61 |

## MATHEMATICAL CONSTANTS IN OCTAL SCALE

| | | |
|---|---|---|
| $\pi = 3.11037\ 552421_8$ | $e = 2.55760\ 521305_8$ | $\gamma = 0.44742\ 147707_8$ |
| $\pi^{-1} = 0.24276\ 301556_8$ | $e^{-1} = 0.27426\ 530661_8$ | $\ln \gamma = -0.43127\ 233602_8$ |
| $\sqrt{\pi} = 1.61337\ 611067_8$ | $\sqrt{e} = 1.51411\ 230704_8$ | $\log_2 \gamma = -0.62573\ 030645_8$ |
| $\ln \pi = 1.11206\ 404435_8$ | $\log_{10} e = 0.33626\ 754251_8$ | $\sqrt{2} = 1.32404\ 746320_8$ |
| $\log_2 \pi = 1.51544\ 163223_8$ | $\log_2 e = 1.34252\ 166245_8$ | $\ln 2 = 0.54271\ 027760_8$ |
| $\sqrt{10} = 3.12305\ 407267_8$ | $\log_2 10 = 3.24464\ 741136_8$ | $\ln 10 = 2.23273\ 067355_8$ |

# APPENDIX 4
# POWERS OF TWO

| $2^n$ | n | $2^{-n}$ |
|---|---|---|
| 1 | 0 | 1.0 |
| 2 | 1 | 0.5 |
| 4 | 2 | 0.25 |
| 8 | 3 | 0.125 |
| 16 | 4 | 0.062 5 |
| 32 | 5 | 0.031 25 |
| 64 | 6 | 0.015 625 |
| 128 | 7 | 0.007 812 5 |
| 256 | 8 | 0.003 906 25 |
| 512 | 9 | 0.001 953 125 |
| 1 024 | 10 | 0.000 976 562 5 |
| 2 048 | 11 | 0.000 488 281 25 |
| 4 096 | 12 | 0.000 244 140 625 |
| 8 192 | 13 | 0.000 122 070 312 5 |
| 16 384 | 14 | 0.000 061 035 156 25 |
| 32 768 | 15 | 0.000 030 517 578 125 |
| 65 536 | 16 | 0.000 015 258 789 062 5 |
| 131 072 | 17 | 0.000 007 629 394 531 25 |
| 262 144 | 18 | 0.000 003 814 697 265 625 |
| 524 288 | 19 | 0.000 001 907 348 632 812 5 |
| 1 048 576 | 20 | 0.000 000 953 674 316 406 25 |
| 2 097 152 | 21 | 0.000 000 476 837 158 203 125 |
| 4 194 304 | 22 | 0.000 000 238 418 579 101 562 5 |
| 8 388 608 | 23 | 0.000 000 119 209 289 550 781 25 |
| 16 777 216 | 24 | 0.000 000 059 604 644 775 390 625 |
| 33 554 432 | 25 | 0.000 000 029 802 322 387 695 312 5 |
| 67 108 864 | 26 | 0.000 000 014 901 161 193 847 656 25 |
| 134 217 728 | 27 | 0.000 000 007 450 580 596 923 828 125 |
| 268 435 456 | 28 | 0.000 000 003 725 290 298 461 914 062 5 |
| 536 870 912 | 29 | 0.000 000 001 862 645 149 230 957 031 25 |
| 1 073 741 824 | 30 | 0.000 000 000 931 322 574 615 478 515 625 |
| 2 147 483 648 | 31 | 0.000 000 000 465 661 287 307 739 257 812 5 |
| 4 294 967 296 | 32 | 0.000 000 000 232 830 643 653 869 628 906 25 |
| 8 589 934 592 | 33 | 0.000 000 000 116 415 321 826 934 814 453 125 |
| 17 179 869 184 | 34 | 0.000 000 000 058 207 660 913 467 407 226 562 5 |
| 34 359 738 368 | 35 | 0.000 000 000 029 103 830 456 733 703 613 281 25 |
| 68 719 476 736 | 36 | 0.000 000 000 014 551 915 228 366 851 806 640 625 |
| 137 438 953 472 | 37 | 0.000 000 000 007 275 957 614 183 425 903 320 312 5 |
| 274 877 906 944 | 38 | 0.000 000 000 003 637 978 807 091 712 951 660 156 25 |
| 549 755 813 888 | 39 | 0.000 000 000 001 818 989 403 545 856 475 830 078 125 |
| 1 099 511 627 776 | 40 | 0.000 000 000 000 909 494 701 772 928 237 915 039 062 5 |
| 2 199 023 255 552 | 41 | 0.000 000 000 000 454 747 350 886 464 118 957 519 531 25 |
| 4 398 046 511 104 | 42 | 0.000 000 000 000 227 373 675 443 232 059 478 759 765 625 |
| 8 796 093 022 208 | 43 | 0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5 |
| 17 592 186 044 416 | 44 | 0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25 |
| 35 184 372 088 832 | 45 | 0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125 |
| 70 368 744 177 664 | 46 | 0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5 |
| 140 737 488 355 328 | 47 | 0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25 |
| 281 474 976 710 656 | 48 | 0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625 |
| 562 949 953 421 312 | 49 | 0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5 |
| 1 125 899 906 842 624 | 50 | 0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25 |
| 2 251 799 813 685 248 | 51 | 0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125 |
| 4 503 599 627 370 496 | 52 | 0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5 |
| 9 007 199 254 740 992 | 53 | 0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25 |
| 18 014 398 509 481 984 | 54 | 0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625 |
| 36 028 797 018 963 968 | 55 | 0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5 |
| 72 057 594 037 927 936 | 56 | 0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25 |
| 144 115 188 075 855 872 | 57 | 0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125 |
| 288 230 376 151 711 744 | 58 | 0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5 |
| 576 460 752 303 423 488 | 59 | 0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25 |
| 1 152 921 504 606 846 976 | 60 | 0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625 |
| 2 305 843 009 213 693 952 | 61 | 0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5 |
| 4 611 686 018 427 387 904 | 62 | 0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25 |
| 9 223 372 036 854 775 808 | 63 | 0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125 |
| 18 446 744 073 709 551 616 | 64 | 0.000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5 |
| 36 893 488 147 419 103 232 | 65 | 0.000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25 |
| 73 786 976 294 838 206 464 | 66 | 0.000 000 000 000 000 000 013 552 527 156 068 805 425 093 160 010 874 271 392 822 265 625 |
| 147 573 952 589 676 412 928 | 67 | 0.000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5 |
| 295 147 905 179 352 825 856 | 68 | 0.000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25 |
| 590 295 810 358 705 651 712 | 69 | 0.000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125 |
| 1 180 591 620 717 411 303 424 | 70 | 0.000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5 |
| 2 361 183 241 434 822 606 848 | 71 | 0.000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25 |
| 4 722 366 482 869 645 213 696 | 72 | 0.000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625 |

# APPENDIX 5

# OCTAL-DECIMAL CONVERSION

## OCTAL-DECIMAL INTEGER CONVERSION TABLE

| 0000 to 0777 (Octal) | 0000 to 0511 (Decimal) |
|---|---|

| Octal | Decimal |
|---|---|
| 10000 | 4096 |
| 20000 | 8192 |
| 30000 | 12288 |
| 40000 | 16384 |
| 50000 | 20480 |
| 60000 | 24576 |
| 70000 | 28672 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0000 | 0000 | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 |
| 0010 | 0008 | 0009 | 0010 | 0011 | 0012 | 0013 | 0014 | 0015 |
| 0020 | 0016 | 0017 | 0018 | 0019 | 0020 | 0021 | 0022 | 0023 |
| 0030 | 0024 | 0025 | 0026 | 0027 | 0028 | 0029 | 0030 | 0031 |
| 0040 | 0032 | 0033 | 0034 | 0035 | 0036 | 0037 | 0038 | 0039 |
| 0050 | 0040 | 0041 | 0042 | 0043 | 0044 | 0045 | 0046 | 0047 |
| 0060 | 0048 | 0049 | 0050 | 0051 | 0052 | 0053 | 0054 | 0055 |
| 0070 | 0056 | 0057 | 0058 | 0059 | 0060 | 0061 | 0062 | 0063 |
| 0100 | 0064 | 0065 | 0066 | 0067 | 0068 | 0069 | 0070 | 0071 |
| 0110 | 0072 | 0073 | 0074 | 0075 | 0076 | 0077 | 0078 | 0079 |
| 0120 | 0080 | 0081 | 0082 | 0083 | 0084 | 0085 | 0086 | 0087 |
| 0130 | 0088 | 0089 | 0090 | 0091 | 0092 | 0093 | 0094 | 0095 |
| 0140 | 0096 | 0097 | 0098 | 0099 | 0100 | 0101 | 0102 | 0103 |
| 0150 | 0104 | 0105 | 0106 | 0107 | 0108 | 0109 | 0110 | 0111 |
| 0160 | 0112 | 0113 | 0114 | 0115 | 0116 | 0117 | 0118 | 0119 |
| 0170 | 0120 | 0121 | 0122 | 0123 | 0124 | 0125 | 0126 | 0127 |
| 0200 | 0128 | 0129 | 0130 | 0131 | 0132 | 0133 | 0134 | 0135 |
| 0210 | 0136 | 0137 | 0138 | 0139 | 0140 | 0141 | 0142 | 0143 |
| 0220 | 0144 | 0145 | 0146 | 0147 | 0148 | 0149 | 0150 | 0151 |
| 0230 | 0152 | 0153 | 0154 | 0155 | 0156 | 0157 | 0158 | 0159 |
| 0240 | 0160 | 0161 | 0162 | 0163 | 0164 | 0165 | 0166 | 0167 |
| 0250 | 0168 | 0169 | 0170 | 0171 | 0172 | 0173 | 0174 | 0175 |
| 0260 | 0176 | 0177 | 0178 | 0179 | 0180 | 0181 | 0182 | 0183 |
| 0270 | 0184 | 0185 | 0186 | 0187 | 0188 | 0189 | 0190 | 0191 |
| 0300 | 0192 | 0193 | 0194 | 0195 | 0196 | 0197 | 0198 | 0199 |
| 0310 | 0200 | 0201 | 0202 | 0203 | 0204 | 0205 | 0206 | 0207 |
| 0320 | 0208 | 0209 | 0210 | 0211 | 0212 | 0213 | 0214 | 0215 |
| 0330 | 0216 | 0217 | 0218 | 0219 | 0220 | 0221 | 0222 | 0223 |
| 0340 | 0224 | 0225 | 0226 | 0227 | 0228 | 0229 | 0230 | 0231 |
| 0350 | 0232 | 0233 | 0234 | 0235 | 0236 | 0237 | 0238 | 0239 |
| 0360 | 0240 | 0241 | 0242 | 0243 | 0244 | 0245 | 0246 | 0247 |
| 0370 | 0248 | 0249 | 0250 | 0251 | 0252 | 0253 | 0254 | 0255 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0400 | 0256 | 0257 | 0258 | 0259 | 0260 | 0261 | 0262 | 0263 |
| 0410 | 0264 | 0265 | 0266 | 0267 | 0268 | 0269 | 0270 | 0271 |
| 0420 | 0272 | 0273 | 0274 | 0275 | 0276 | 0277 | 0278 | 0279 |
| 0430 | 0280 | 0281 | 0282 | 0283 | 0284 | 0285 | 0286 | 0287 |
| 0440 | 0288 | 0289 | 0290 | 0291 | 0292 | 0293 | 0294 | 0295 |
| 0450 | 0296 | 0297 | 0298 | 0299 | 0300 | 0301 | 0302 | 0303 |
| 0460 | 0304 | 0305 | 0306 | 0307 | 0308 | 0309 | 0310 | 0311 |
| 0470 | 0312 | 0313 | 0314 | 0315 | 0316 | 0317 | 0318 | 0319 |
| 0500 | 0320 | 0321 | 0322 | 0323 | 0324 | 0325 | 0326 | 0327 |
| 0510 | 0328 | 0329 | 0330 | 0331 | 0332 | 0333 | 0334 | 0335 |
| 0520 | 0336 | 0337 | 0338 | 0339 | 0340 | 0341 | 0342 | 0343 |
| 0530 | 0344 | 0345 | 0346 | 0347 | 0348 | 0349 | 0350 | 0351 |
| 0540 | 0352 | 0353 | 0354 | 0355 | 0356 | 0357 | 0358 | 0359 |
| 0550 | 0360 | 0361 | 0362 | 0363 | 0364 | 0365 | 0366 | 0367 |
| 0560 | 0368 | 0369 | 0370 | 0371 | 0372 | 0373 | 0374 | 0375 |
| 0570 | 0376 | 0377 | 0378 | 0379 | 0380 | 0381 | 0382 | 0383 |
| 0600 | 0384 | 0385 | 0386 | 0387 | 0388 | 0389 | 0390 | 0391 |
| 0610 | 0392 | 0393 | 0394 | 0395 | 0396 | 0397 | 0398 | 0399 |
| 0620 | 0400 | 0401 | 0402 | 0403 | 0404 | 0405 | 0406 | 0407 |
| 0630 | 0408 | 0409 | 0410 | 0411 | 0412 | 0413 | 0414 | 0415 |
| 0640 | 0416 | 0417 | 0418 | 0419 | 0420 | 0421 | 0422 | 0423 |
| 0650 | 0424 | 0425 | 0426 | 0427 | 0428 | 0429 | 0430 | 0431 |
| 0660 | 0432 | 0433 | 0434 | 0435 | 0436 | 0437 | 0438 | 0439 |
| 0670 | 0440 | 0441 | 0442 | 0443 | 0444 | 0445 | 0446 | 0447 |
| 0700 | 0448 | 0449 | 0450 | 0451 | 0452 | 0453 | 0454 | 0455 |
| 0710 | 0456 | 0457 | 0458 | 0459 | 0460 | 0461 | 0462 | 0463 |
| 0720 | 0464 | 0465 | 0466 | 0467 | 0468 | 0469 | 0470 | 0471 |
| 0730 | 0472 | 0473 | 0474 | 0475 | 0476 | 0477 | 0478 | 0479 |
| 0740 | 0480 | 0481 | 0482 | 0483 | 0484 | 0485 | 0486 | 0487 |
| 0750 | 0488 | 0489 | 0490 | 0491 | 0492 | 0493 | 0494 | 0495 |
| 0760 | 0496 | 0497 | 0498 | 0499 | 0500 | 0501 | 0502 | 0503 |
| 0770 | 0504 | 0505 | 0506 | 0507 | 0508 | 0509 | 0510 | 0511 |

| 1000 to 1777 (Octal) | 0512 to 1023 (Decimal) |
|---|---|

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1000 | 0512 | 0513 | 0514 | 0515 | 0516 | 0517 | 0518 | 0519 |
| 1010 | 0520 | 0521 | 0522 | 0523 | 0524 | 0525 | 0526 | 0527 |
| 1020 | 0528 | 0529 | 0530 | 0531 | 0532 | 0533 | 0534 | 0535 |
| 1030 | 0536 | 0537 | 0538 | 0539 | 0540 | 0541 | 0542 | 0543 |
| 1040 | 0544 | 0545 | 0546 | 0547 | 0548 | 0549 | 0550 | 0551 |
| 1050 | 0552 | 0553 | 0554 | 0555 | 0556 | 0557 | 0558 | 0559 |
| 1060 | 0560 | 0561 | 0562 | 0563 | 0564 | 0565 | 0566 | 0567 |
| 1070 | 0568 | 0569 | 0570 | 0571 | 0572 | 0573 | 0574 | 0575 |
| 1100 | 0576 | 0577 | 0578 | 0579 | 0580 | 0581 | 0582 | 0583 |
| 1110 | 0584 | 0585 | 0586 | 0587 | 0588 | 0589 | 0590 | 0591 |
| 1120 | 0592 | 0593 | 0594 | 0595 | 0596 | 0597 | 0598 | 0599 |
| 1130 | 0600 | 0601 | 0602 | 0603 | 0604 | 0605 | 0606 | 0607 |
| 1140 | 0608 | 0609 | 0610 | 0611 | 0612 | 0613 | 0614 | 0615 |
| 1150 | 0616 | 0617 | 0618 | 0619 | 0620 | 0621 | 0622 | 0623 |
| 1160 | 0624 | 0625 | 0626 | 0627 | 0628 | 0629 | 0630 | 0631 |
| 1170 | 0632 | 0633 | 0634 | 0635 | 0636 | 0637 | 0638 | 0639 |
| 1200 | 0640 | 0641 | 0642 | 0643 | 0644 | 0645 | 0646 | 0647 |
| 1210 | 0648 | 0649 | 0650 | 0651 | 0652 | 0653 | 0654 | 0655 |
| 1220 | 0656 | 0657 | 0658 | 0659 | 0660 | 0661 | 0662 | 0663 |
| 1230 | 0664 | 0665 | 0666 | 0667 | 0668 | 0669 | 0670 | 0671 |
| 1240 | 0672 | 0673 | 0674 | 0675 | 0676 | 0677 | 0678 | 0679 |
| 1250 | 0680 | 0681 | 0682 | 0683 | 0684 | 0685 | 0686 | 0687 |
| 1260 | 0688 | 0689 | 0690 | 0691 | 0692 | 0693 | 0694 | 0695 |
| 1270 | 0696 | 0697 | 0698 | 0699 | 0700 | 0701 | 0702 | 0703 |
| 1300 | 0704 | 0705 | 0706 | 0707 | 0708 | 0709 | 0710 | 0711 |
| 1310 | 0712 | 0713 | 0714 | 0715 | 0716 | 0717 | 0718 | 0719 |
| 1320 | 0720 | 0721 | 0722 | 0723 | 0724 | 0725 | 0726 | 0727 |
| 1330 | 0728 | 0729 | 0730 | 0731 | 0732 | 0733 | 0734 | 0735 |
| 1340 | 0736 | 0737 | 0738 | 0739 | 0740 | 0741 | 0742 | 0743 |
| 1350 | 0744 | 0745 | 0746 | 0747 | 0748 | 0749 | 0750 | 0751 |
| 1360 | 0752 | 0753 | 0754 | 0755 | 0756 | 0757 | 0758 | 0759 |
| 1370 | 0760 | 0761 | 0762 | 0763 | 0764 | 0765 | 0766 | 0767 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1400 | 0768 | 0769 | 0770 | 0771 | 0772 | 0773 | 0774 | 0775 |
| 1410 | 0776 | 0777 | 0778 | 0779 | 0780 | 0781 | 0782 | 0783 |
| 1420 | 0784 | 0785 | 0786 | 0787 | 0788 | 0789 | 0790 | 0791 |
| 1430 | 0792 | 0793 | 0794 | 0795 | 0796 | 0797 | 0798 | 0799 |
| 1440 | 0800 | 0801 | 0802 | 0803 | 0804 | 0805 | 0806 | 0807 |
| 1450 | 0808 | 0809 | 0810 | 0811 | 0812 | 0813 | 0814 | 0815 |
| 1460 | 0816 | 0817 | 0818 | 0819 | 0820 | 0821 | 0822 | 0823 |
| 1470 | 0824 | 0825 | 0826 | 0827 | 0828 | 0829 | 0830 | 0831 |
| 1500 | 0832 | 0833 | 0834 | 0835 | 0836 | 0837 | 0838 | 0839 |
| 1510 | 0840 | 0841 | 0842 | 0843 | 0844 | 0845 | 0846 | 0847 |
| 1520 | 0848 | 0849 | 0850 | 0851 | 0852 | 0853 | 0854 | 0855 |
| 1530 | 0856 | 0857 | 0858 | 0859 | 0860 | 0861 | 0862 | 0863 |
| 1540 | 0864 | 0865 | 0866 | 0867 | 0868 | 0869 | 0870 | 0871 |
| 1550 | 0872 | 0873 | 0874 | 0875 | 0876 | 0877 | 0878 | 0879 |
| 1560 | 0880 | 0881 | 0882 | 0883 | 0884 | 0885 | 0886 | 0887 |
| 1570 | 0888 | 0889 | 0890 | 0891 | 0892 | 0893 | 0894 | 0895 |
| 1600 | 0896 | 0897 | 0898 | 0899 | 0900 | 0901 | 0902 | 0903 |
| 1610 | 0904 | 0905 | 0906 | 0907 | 0908 | 0909 | 0910 | 0911 |
| 1620 | 0912 | 0913 | 0914 | 0915 | 0916 | 0917 | 0918 | 0919 |
| 1630 | 0920 | 0921 | 0922 | 0923 | 0924 | 0925 | 0926 | 0927 |
| 1640 | 0928 | 0929 | 0930 | 0931 | 0932 | 0933 | 0934 | 0935 |
| 1650 | 0936 | 0937 | 0938 | 0939 | 0940 | 0941 | 0942 | 0943 |
| 1660 | 0944 | 0945 | 0946 | 0947 | 0948 | 0949 | 0950 | 0951 |
| 1670 | 0952 | 0953 | 0954 | 0955 | 0956 | 0957 | 0958 | 0959 |
| 1700 | 0960 | 0961 | 0962 | 0963 | 0964 | 0965 | 0966 | 0967 |
| 1710 | 0968 | 0969 | 0970 | 0971 | 0972 | 0973 | 0974 | 0975 |
| 1720 | 0976 | 0977 | 0978 | 0979 | 0980 | 0981 | 0982 | 0983 |
| 1730 | 0984 | 0985 | 0986 | 0987 | 0988 | 0989 | 0990 | 0991 |
| 1740 | 0992 | 0993 | 0994 | 0995 | 0996 | 0997 | 0998 | 0999 |
| 1750 | 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 |
| 1760 | 1008 | 1009 | 1010 | 1011 | 1012 | 1013 | 1014 | 1015 |
| 1770 | 1016 | 1017 | 1018 | 1019 | 1020 | 1021 | 1022 | 1023 |

# OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 2000 | 1024 | 1025 | 1026 | 1027 | 1028 | 1029 | 1030 | 1031 |
| 2010 | 1032 | 1033 | 1034 | 1035 | 1036 | 1037 | 1038 | 1039 |
| 2020 | 1040 | 1041 | 1042 | 1043 | 1044 | 1045 | 1046 | 1047 |
| 2030 | 1048 | 1049 | 1050 | 1051 | 1052 | 1053 | 1054 | 1055 |
| 2040 | 1056 | 1057 | 1058 | 1059 | 1060 | 1061 | 1062 | 1063 |
| 2050 | 1064 | 1065 | 1066 | 1067 | 1068 | 1069 | 1070 | 1071 |
| 2060 | 1072 | 1073 | 1074 | 1075 | 1076 | 1077 | 1078 | 1079 |
| 2070 | 1080 | 1081 | 1082 | 1083 | 1084 | 1085 | 1086 | 1087 |
| 2100 | 1088 | 1089 | 1090 | 1091 | 1092 | 1093 | 1094 | 1095 |
| 2110 | 1096 | 1097 | 1098 | 1099 | 1100 | 1101 | 1102 | 1103 |
| 2120 | 1104 | 1105 | 1106 | 1107 | 1108 | 1109 | 1110 | 1111 |
| 2130 | 1112 | 1113 | 1114 | 1115 | 1116 | 1117 | 1118 | 1119 |
| 2140 | 1120 | 1121 | 1122 | 1123 | 1124 | 1125 | 1126 | 1127 |
| 2150 | 1128 | 1129 | 1130 | 1131 | 1132 | 1133 | 1134 | 1135 |
| 2160 | 1136 | 1137 | 1138 | 1139 | 1140 | 1141 | 1142 | 1143 |
| 2170 | 1144 | 1145 | 1146 | 1147 | 1148 | 1149 | 1150 | 1151 |
| 2200 | 1152 | 1153 | 1154 | 1155 | 1156 | 1157 | 1158 | 1159 |
| 2210 | 1160 | 1161 | 1162 | 1163 | 1164 | 1165 | 1166 | 1167 |
| 2220 | 1168 | 1169 | 1170 | 1171 | 1172 | 1173 | 1174 | 1175 |
| 2230 | 1176 | 1177 | 1178 | 1179 | 1180 | 1181 | 1182 | 1183 |
| 2240 | 1184 | 1185 | 1186 | 1187 | 1188 | 1189 | 1190 | 1191 |
| 2250 | 1192 | 1193 | 1194 | 1195 | 1196 | 1197 | 1198 | 1199 |
| 2260 | 1200 | 1201 | 1202 | 1203 | 1204 | 1205 | 1206 | 1207 |
| 2270 | 1208 | 1209 | 1210 | 1211 | 1212 | 1213 | 1214 | 1215 |
| 2300 | 1216 | 1217 | 1218 | 1219 | 1220 | 1221 | 1222 | 1223 |
| 2310 | 1224 | 1225 | 1226 | 1227 | 1228 | 1229 | 1230 | 1231 |
| 2320 | 1232 | 1233 | 1234 | 1235 | 1236 | 1237 | 1238 | 1239 |
| 2330 | 1240 | 1241 | 1242 | 1243 | 1244 | 1245 | 1246 | 1247 |
| 2340 | 1248 | 1249 | 1250 | 1251 | 1252 | 1253 | 1254 | 1255 |
| 2350 | 1256 | 1257 | 1258 | 1259 | 1260 | 1261 | 1262 | 1263 |
| 2360 | 1264 | 1265 | 1266 | 1267 | 1268 | 1269 | 1270 | 1271 |
| 2370 | 1272 | 1273 | 1274 | 1275 | 1276 | 1277 | 1278 | 1279 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 2400 | 1280 | 1281 | 1282 | 1283 | 1284 | 1285 | 1286 | 1287 |
| 2410 | 1288 | 1289 | 1290 | 1291 | 1292 | 1293 | 1294 | 1295 |
| 2420 | 1296 | 1297 | 1298 | 1299 | 1300 | 1301 | 1302 | 1303 |
| 2430 | 1304 | 1305 | 1306 | 1307 | 1308 | 1309 | 1310 | 1311 |
| 2440 | 1312 | 1313 | 1314 | 1315 | 1316 | 1317 | 1318 | 1319 |
| 2450 | 1320 | 1321 | 1322 | 1323 | 1324 | 1325 | 1326 | 1327 |
| 2460 | 1328 | 1329 | 1330 | 1331 | 1332 | 1333 | 1334 | 1335 |
| 2470 | 1336 | 1337 | 1338 | 1339 | 1340 | 1341 | 1342 | 1343 |
| 2500 | 1344 | 1345 | 1346 | 1347 | 1348 | 1349 | 1350 | 1351 |
| 2510 | 1352 | 1353 | 1354 | 1355 | 1356 | 1357 | 1358 | 1359 |
| 2520 | 1360 | 1361 | 1362 | 1363 | 1364 | 1365 | 1366 | 1367 |
| 2530 | 1368 | 1369 | 1370 | 1371 | 1372 | 1373 | 1374 | 1375 |
| 2540 | 1376 | 1377 | 1378 | 1379 | 1380 | 1381 | 1382 | 1383 |
| 2550 | 1384 | 1385 | 1386 | 1387 | 1388 | 1389 | 1390 | 1391 |
| 2560 | 1392 | 1393 | 1394 | 1395 | 1396 | 1397 | 1398 | 1399 |
| 2570 | 1400 | 1401 | 1402 | 1403 | 1404 | 1405 | 1406 | 1407 |
| 2600 | 1408 | 1409 | 1410 | 1411 | 1412 | 1413 | 1414 | 1415 |
| 2610 | 1416 | 1417 | 1418 | 1419 | 1420 | 1421 | 1422 | 1423 |
| 2620 | 1424 | 1425 | 1426 | 1427 | 1428 | 1429 | 1430 | 1431 |
| 2630 | 1432 | 1433 | 1434 | 1435 | 1436 | 1437 | 1438 | 1439 |
| 2640 | 1440 | 1441 | 1442 | 1443 | 1444 | 1445 | 1446 | 1447 |
| 2650 | 1448 | 1449 | 1450 | 1451 | 1452 | 1453 | 1454 | 1455 |
| 2660 | 1456 | 1457 | 1458 | 1459 | 1460 | 1461 | 1462 | 1463 |
| 2670 | 1464 | 1465 | 1466 | 1467 | 1468 | 1469 | 1470 | 1471 |
| 2700 | 1472 | 1473 | 1474 | 1475 | 1476 | 1477 | 1478 | 1479 |
| 2710 | 1480 | 1481 | 1482 | 1483 | 1484 | 1485 | 1486 | 1487 |
| 2720 | 1488 | 1489 | 1490 | 1491 | 1492 | 1493 | 1494 | 1495 |
| 2730 | 1496 | 1497 | 1498 | 1499 | 1500 | 1501 | 1502 | 1503 |
| 2740 | 1504 | 1505 | 1506 | 1507 | 1508 | 1509 | 1510 | 1511 |
| 2750 | 1512 | 1513 | 1514 | 1515 | 1516 | 1517 | 1518 | 1519 |
| 2760 | 1520 | 1521 | 1522 | 1523 | 1524 | 1525 | 1526 | 1527 |
| 2770 | 1528 | 1529 | 1530 | 1531 | 1532 | 1533 | 1534 | 1535 |

| 2000 to 2777 (Octal) | 1024 to 1535 (Decimal) |
|---|---|

| Octal | Decimal |
|---|---|
| 10000 - | 4096 |
| 20000 - | 8192 |
| 30000 - | 12288 |
| 40000 - | 16384 |
| 50000 - | 20480 |
| 60000 - | 24576 |
| 70000 - | 28672 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 3000 | 1536 | 1537 | 1538 | 1539 | 1540 | 1541 | 1542 | 1543 |
| 3010 | 1544 | 1545 | 1546 | 1547 | 1548 | 1549 | 1550 | 1551 |
| 3020 | 1552 | 1553 | 1554 | 1555 | 1556 | 1557 | 1558 | 1559 |
| 3030 | 1560 | 1561 | 1562 | 1563 | 1564 | 1565 | 1566 | 1567 |
| 3040 | 1568 | 1569 | 1570 | 1571 | 1572 | 1573 | 1574 | 1575 |
| 3050 | 1576 | 1577 | 1578 | 1579 | 1580 | 1581 | 1582 | 1583 |
| 3060 | 1584 | 1585 | 1586 | 1587 | 1588 | 1589 | 1590 | 1591 |
| 3070 | 1592 | 1593 | 1594 | 1595 | 1596 | 1597 | 1598 | 1599 |
| 3100 | 1600 | 1601 | 1602 | 1603 | 1604 | 1605 | 1606 | 1607 |
| 3110 | 1608 | 1609 | 1610 | 1611 | 1612 | 1613 | 1614 | 1615 |
| 3120 | 1616 | 1617 | 1618 | 1619 | 1620 | 1621 | 1622 | 1623 |
| 3130 | 1624 | 1625 | 1626 | 1627 | 1628 | 1629 | 1630 | 1631 |
| 3140 | 1632 | 1633 | 1634 | 1635 | 1636 | 1637 | 1638 | 1639 |
| 3150 | 1640 | 1641 | 1642 | 1643 | 1644 | 1645 | 1646 | 1647 |
| 3160 | 1648 | 1649 | 1650 | 1651 | 1652 | 1653 | 1654 | 1655 |
| 3170 | 1656 | 1657 | 1658 | 1659 | 1660 | 1661 | 1662 | 1663 |
| 3200 | 1664 | 1665 | 1666 | 1667 | 1668 | 1669 | 1670 | 1671 |
| 3210 | 1672 | 1673 | 1674 | 1675 | 1676 | 1677 | 1678 | 1679 |
| 3220 | 1680 | 1681 | 1682 | 1683 | 1684 | 1685 | 1686 | 1687 |
| 3230 | 1688 | 1689 | 1690 | 1691 | 1692 | 1693 | 1694 | 1695 |
| 3240 | 1696 | 1697 | 1698 | 1699 | 1700 | 1701 | 1702 | 1703 |
| 3250 | 1704 | 1705 | 1706 | 1707 | 1708 | 1709 | 1710 | 1711 |
| 3260 | 1712 | 1713 | 1714 | 1715 | 1716 | 1717 | 1718 | 1719 |
| 3270 | 1720 | 1721 | 1722 | 1723 | 1724 | 1725 | 1726 | 1727 |
| 3300 | 1728 | 1729 | 1730 | 1731 | 1732 | 1733 | 1734 | 1735 |
| 3310 | 1736 | 1737 | 1738 | 1739 | 1740 | 1741 | 1742 | 1743 |
| 3320 | 1744 | 1745 | 1746 | 1747 | 1748 | 1749 | 1750 | 1751 |
| 3330 | 1752 | 1753 | 1754 | 1755 | 1756 | 1757 | 1758 | 1759 |
| 3340 | 1760 | 1761 | 1762 | 1763 | 1764 | 1765 | 1766 | 1767 |
| 3350 | 1768 | 1769 | 1770 | 1771 | 1772 | 1773 | 1774 | 1775 |
| 3360 | 1776 | 1777 | 1778 | 1779 | 1780 | 1781 | 1782 | 1783 |
| 3370 | 1784 | 1785 | 1786 | 1787 | 1788 | 1789 | 1790 | 1791 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 3400 | 1792 | 1793 | 1794 | 1795 | 1796 | 1797 | 1798 | 1799 |
| 3410 | 1800 | 1801 | 1802 | 1803 | 1804 | 1805 | 1806 | 1807 |
| 3420 | 1808 | 1809 | 1810 | 1811 | 1812 | 1813 | 1814 | 1815 |
| 3430 | 1816 | 1817 | 1818 | 1819 | 1820 | 1821 | 1822 | 1823 |
| 3440 | 1824 | 1825 | 1826 | 1827 | 1828 | 1829 | 1830 | 1831 |
| 3450 | 1832 | 1833 | 1834 | 1835 | 1836 | 1837 | 1838 | 1839 |
| 3460 | 1840 | 1841 | 1842 | 1843 | 1844 | 1845 | 1846 | 1847 |
| 3470 | 1848 | 1849 | 1850 | 1851 | 1852 | 1853 | 1854 | 1855 |
| 3500 | 1856 | 1857 | 1858 | 1859 | 1860 | 1861 | 1862 | 1863 |
| 3510 | 1864 | 1865 | 1866 | 1867 | 1868 | 1869 | 1870 | 1871 |
| 3520 | 1872 | 1873 | 1874 | 1875 | 1876 | 1877 | 1878 | 1879 |
| 3530 | 1880 | 1881 | 1882 | 1883 | 1884 | 1885 | 1886 | 1887 |
| 3540 | 1888 | 1889 | 1890 | 1891 | 1892 | 1893 | 1894 | 1895 |
| 3550 | 1896 | 1897 | 1898 | 1899 | 1900 | 1901 | 1902 | 1903 |
| 3560 | 1904 | 1905 | 1906 | 1907 | 1908 | 1909 | 1910 | 1911 |
| 3570 | 1912 | 1913 | 1914 | 1915 | 1916 | 1917 | 1918 | 1919 |
| 3600 | 1920 | 1921 | 1922 | 1923 | 1924 | 1925 | 1926 | 1927 |
| 3610 | 1928 | 1929 | 1930 | 1931 | 1932 | 1933 | 1934 | 1935 |
| 3620 | 1936 | 1937 | 1938 | 1939 | 1940 | 1941 | 1942 | 1943 |
| 3630 | 1944 | 1945 | 1946 | 1947 | 1948 | 1949 | 1950 | 1951 |
| 3640 | 1952 | 1953 | 1954 | 1955 | 1956 | 1957 | 1958 | 1959 |
| 3650 | 1960 | 1961 | 1962 | 1963 | 1964 | 1965 | 1966 | 1967 |
| 3660 | 1968 | 1969 | 1970 | 1971 | 1972 | 1973 | 1974 | 1975 |
| 3670 | 1976 | 1977 | 1978 | 1979 | 1980 | 1981 | 1982 | 1983 |
| 3700 | 1984 | 1985 | 1986 | 1987 | 1988 | 1989 | 1990 | 1991 |
| 3710 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 |
| 3720 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 |
| 3730 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 |
| 3740 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 |
| 3750 | 2024 | 2025 | 2026 | 2027 | 2028 | 2029 | 2030 | 2031 |
| 3760 | 2032 | 2033 | 2034 | 2035 | 2036 | 2037 | 2038 | 2039 |
| 3770 | 2040 | 2041 | 2042 | 2043 | 2044 | 2045 | 2046 | 2047 |

| 3000 to 3777 (Octal) | 1536 to 2047 (Decimal) |
|---|---|

# OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

| 4000 to 4777 (Octal) | 2048 to 2559 (Decimal) |
|---|---|

| Octal | Decimal |
|---|---|
| 10000 - | 4096 |
| 20000 - | 8192 |
| 30000 - | 12288 |
| 40000 - | 16384 |
| 50000 - | 20480 |
| 60000 - | 24576 |
| 70000 - | 28672 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 4000 | 2048 | 2049 | 2050 | 2051 | 2052 | 2053 | 2054 | 2055 |
| 4010 | 2056 | 2057 | 2058 | 2059 | 2060 | 2061 | 2062 | 2063 |
| 4020 | 2064 | 2065 | 2066 | 2067 | 2068 | 2069 | 2070 | 2071 |
| 4030 | 2072 | 2073 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 |
| 4040 | 2080 | 2081 | 2082 | 2083 | 2084 | 2085 | 2086 | 2087 |
| 4050 | 2088 | 2089 | 2090 | 2091 | 2092 | 2093 | 2094 | 2095 |
| 4060 | 2096 | 2097 | 2098 | 2099 | 2100 | 2101 | 2102 | 2103 |
| 4070 | 2104 | 2105 | 2106 | 2107 | 2108 | 2109 | 2110 | 2111 |
| 4100 | 2112 | 2113 | 2114 | 2115 | 2116 | 2117 | 2118 | 2119 |
| 4110 | 2120 | 2121 | 2122 | 2123 | 2124 | 2125 | 2126 | 2127 |
| 4120 | 2128 | 2129 | 2130 | 2131 | 2132 | 2133 | 2134 | 2135 |
| 4130 | 2136 | 2137 | 2138 | 2139 | 2140 | 2141 | 2142 | 2143 |
| 4140 | 2144 | 2145 | 2146 | 2147 | 2148 | 2149 | 2150 | 2151 |
| 4150 | 2152 | 2153 | 2154 | 2155 | 2156 | 2157 | 2158 | 2159 |
| 4160 | 2160 | 2161 | 2162 | 2163 | 2164 | 2165 | 2166 | 2167 |
| 4170 | 2168 | 2169 | 2170 | 2171 | 2172 | 2173 | 2174 | 2175 |
| 4200 | 2176 | 2177 | 2178 | 2179 | 2180 | 2181 | 2182 | 2183 |
| 4210 | 2184 | 2185 | 2186 | 2187 | 2188 | 2189 | 2190 | 2191 |
| 4220 | 2192 | 2193 | 2194 | 2195 | 2196 | 2197 | 2198 | 2199 |
| 4230 | 2200 | 2201 | 2202 | 2203 | 2204 | 2205 | 2206 | 2207 |
| 4240 | 2208 | 2209 | 2210 | 2211 | 2212 | 2213 | 2214 | 2215 |
| 4250 | 2216 | 2217 | 2218 | 2219 | 2220 | 2221 | 2222 | 2223 |
| 4260 | 2224 | 2225 | 2226 | 2227 | 2228 | 2229 | 2230 | 2231 |
| 4270 | 2232 | 2233 | 2234 | 2235 | 2236 | 2237 | 2238 | 2239 |
| 4300 | 2240 | 2241 | 2242 | 2243 | 2244 | 2245 | 2246 | 2247 |
| 4310 | 2248 | 2249 | 2250 | 2251 | 2252 | 2253 | 2254 | 2255 |
| 4320 | 2256 | 2257 | 2258 | 2259 | 2260 | 2261 | 2262 | 2263 |
| 4330 | 2264 | 2265 | 2266 | 2267 | 2268 | 2269 | 2270 | 2271 |
| 4340 | 2272 | 2273 | 2274 | 2275 | 2276 | 2277 | 2278 | 2279 |
| 4350 | 2280 | 2281 | 2282 | 2283 | 2284 | 2285 | 2286 | 2287 |
| 4360 | 2288 | 2289 | 2290 | 2291 | 2292 | 2293 | 2294 | 2295 |
| 4370 | 2296 | 2297 | 2298 | 2299 | 2300 | 2301 | 2302 | 2303 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 4400 | 2304 | 2305 | 2306 | 2307 | 2308 | 2309 | 2310 | 2311 |
| 4410 | 2312 | 2313 | 2314 | 2315 | 2316 | 2317 | 2318 | 2319 |
| 4420 | 2320 | 2321 | 2322 | 2323 | 2324 | 2325 | 2326 | 2327 |
| 4430 | 2328 | 2329 | 2330 | 2331 | 2332 | 2333 | 2334 | 2335 |
| 4440 | 2336 | 2337 | 2338 | 2339 | 2340 | 2341 | 2342 | 2343 |
| 4450 | 2344 | 2345 | 2346 | 2347 | 2348 | 2349 | 2350 | 2351 |
| 4460 | 2352 | 2353 | 2354 | 2355 | 2356 | 2357 | 2358 | 2359 |
| 4470 | 2360 | 2361 | 2362 | 2363 | 2364 | 2365 | 2366 | 2367 |
| 4500 | 2368 | 2369 | 2370 | 2371 | 2372 | 2373 | 2374 | 2375 |
| 4510 | 2376 | 2377 | 2378 | 2379 | 2380 | 2381 | 2382 | 2383 |
| 4520 | 2384 | 2385 | 2386 | 2387 | 2388 | 2389 | 2390 | 2391 |
| 4530 | 2392 | 2393 | 2394 | 2395 | 2396 | 2397 | 2398 | 2399 |
| 4540 | 2400 | 2401 | 2402 | 2403 | 2404 | 2405 | 2406 | 2407 |
| 4550 | 2408 | 2409 | 2410 | 2411 | 2412 | 2413 | 2414 | 2415 |
| 4560 | 2416 | 2417 | 2418 | 2419 | 2420 | 2421 | 2422 | 2423 |
| 4570 | 2424 | 2425 | 2426 | 2427 | 2428 | 2429 | 2430 | 2431 |
| 4600 | 2432 | 2433 | 2434 | 2435 | 2436 | 2437 | 2438 | 2439 |
| 4610 | 2440 | 2441 | 2442 | 2443 | 2444 | 2445 | 2446 | 2447 |
| 4620 | 2448 | 2449 | 2450 | 2451 | 2452 | 2453 | 2454 | 2455 |
| 4630 | 2456 | 2457 | 2458 | 2459 | 2460 | 2461 | 2462 | 2463 |
| 4640 | 2464 | 2465 | 2466 | 2467 | 2468 | 2469 | 2470 | 2471 |
| 4650 | 2472 | 2473 | 2474 | 2475 | 2476 | 2477 | 2478 | 2479 |
| 4660 | 2480 | 2481 | 2482 | 2483 | 2484 | 2485 | 2486 | 2487 |
| 4670 | 2488 | 2489 | 2490 | 2491 | 2492 | 2493 | 2494 | 2495 |
| 4700 | 2496 | 2497 | 2498 | 2499 | 2500 | 2501 | 2502 | 2503 |
| 4710 | 2504 | 2505 | 2506 | 2507 | 2508 | 2509 | 2510 | 2511 |
| 4720 | 2512 | 2513 | 2514 | 2515 | 2516 | 2517 | 2518 | 2519 |
| 4730 | 2520 | 2521 | 2522 | 2523 | 2524 | 2525 | 2526 | 2527 |
| 4740 | 2528 | 2529 | 2530 | 2531 | 2532 | 2533 | 2534 | 2535 |
| 4750 | 2536 | 2537 | 2538 | 2539 | 2540 | 2541 | 2542 | 2543 |
| 4760 | 2544 | 2545 | 2546 | 2547 | 2548 | 2549 | 2550 | 2551 |
| 4770 | 2552 | 2553 | 2554 | 2555 | 2556 | 2557 | 2558 | 2559 |

| 5000 to 5777 (Octal) | 2560 to 3071 (Decimal) |
|---|---|

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 5000 | 2560 | 2561 | 2562 | 2563 | 2564 | 2565 | 2566 | 2567 |
| 5010 | 2568 | 2569 | 2570 | 2571 | 2572 | 2573 | 2574 | 2575 |
| 5020 | 2576 | 2577 | 2578 | 2579 | 2580 | 2581 | 2582 | 2583 |
| 5030 | 2584 | 2585 | 2586 | 2587 | 2588 | 2589 | 2590 | 2591 |
| 5040 | 2592 | 2593 | 2594 | 2595 | 2596 | 2597 | 2598 | 2599 |
| 5050 | 2600 | 2601 | 2602 | 2603 | 2604 | 2605 | 2606 | 2607 |
| 5060 | 2608 | 2609 | 2610 | 2611 | 2612 | 2613 | 2614 | 2615 |
| 5070 | 2616 | 2617 | 2618 | 2619 | 2620 | 2621 | 2622 | 2623 |
| 5100 | 2624 | 2625 | 2626 | 2627 | 2628 | 2629 | 2630 | 2631 |
| 5110 | 2632 | 2633 | 2634 | 2635 | 2636 | 2637 | 2638 | 2639 |
| 5120 | 2640 | 2641 | 2642 | 2643 | 2644 | 2645 | 2646 | 2647 |
| 5130 | 2648 | 2649 | 2650 | 2651 | 2652 | 2653 | 2654 | 2655 |
| 5140 | 2656 | 2657 | 2658 | 2659 | 2660 | 2661 | 2662 | 2663 |
| 5150 | 2664 | 2665 | 2666 | 2667 | 2668 | 2669 | 2670 | 2671 |
| 5160 | 2672 | 2673 | 2674 | 2675 | 2676 | 2677 | 2678 | 2679 |
| 5170 | 2680 | 2681 | 2682 | 2683 | 2684 | 2685 | 2686 | 2687 |
| 5200 | 2688 | 2689 | 2690 | 2691 | 2692 | 2693 | 2694 | 2695 |
| 5210 | 2696 | 2697 | 2698 | 2699 | 2700 | 2701 | 2702 | 2703 |
| 5220 | 2704 | 2705 | 2706 | 2707 | 2708 | 2709 | 2710 | 2711 |
| 5230 | 2712 | 2713 | 2714 | 2715 | 2716 | 2717 | 2718 | 2719 |
| 5240 | 2720 | 2721 | 2722 | 2723 | 2724 | 2725 | 2726 | 2727 |
| 5250 | 2728 | 2729 | 2730 | 2731 | 2732 | 2733 | 2734 | 2735 |
| 5260 | 2736 | 2737 | 2738 | 2739 | 2740 | 2741 | 2742 | 2743 |
| 5270 | 2744 | 2745 | 2746 | 2747 | 2748 | 2749 | 2750 | 2751 |
| 5300 | 2752 | 2753 | 2754 | 2755 | 2756 | 2757 | 2758 | 2759 |
| 5310 | 2760 | 2761 | 2762 | 2763 | 2764 | 2765 | 2766 | 2767 |
| 5320 | 2768 | 2769 | 2770 | 2771 | 2772 | 2773 | 2774 | 2775 |
| 5330 | 2776 | 2777 | 2778 | 2779 | 2780 | 2781 | 2782 | 2783 |
| 5340 | 2784 | 2785 | 2786 | 2787 | 2788 | 2789 | 2790 | 2791 |
| 5350 | 2792 | 2793 | 2794 | 2795 | 2796 | 2797 | 2798 | 2799 |
| 5360 | 2800 | 2801 | 2802 | 2803 | 2804 | 2805 | 2806 | 2807 |
| 5370 | 2808 | 2809 | 2810 | 2811 | 2812 | 2813 | 2814 | 2815 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 5400 | 2816 | 2817 | 2818 | 2819 | 2820 | 2821 | 2822 | 2823 |
| 5410 | 2824 | 2825 | 2826 | 2827 | 2828 | 2829 | 2830 | 2831 |
| 5420 | 2832 | 2833 | 2834 | 2835 | 2836 | 2837 | 2838 | 2839 |
| 5430 | 2840 | 2841 | 2842 | 2843 | 2844 | 2845 | 2846 | 2847 |
| 5440 | 2848 | 2849 | 2850 | 2851 | 2852 | 2853 | 2854 | 2855 |
| 5450 | 2856 | 2857 | 2858 | 2859 | 2860 | 2861 | 2862 | 2863 |
| 5460 | 2864 | 2865 | 2866 | 2867 | 2868 | 2869 | 2870 | 2871 |
| 5470 | 2872 | 2873 | 2874 | 2875 | 2876 | 2877 | 2878 | 2879 |
| 5500 | 2880 | 2881 | 2882 | 2883 | 2884 | 2885 | 2886 | 2887 |
| 5510 | 2888 | 2889 | 2890 | 2891 | 2892 | 2893 | 2894 | 2895 |
| 5520 | 2896 | 2897 | 2898 | 2899 | 2900 | 2901 | 2902 | 2903 |
| 5530 | 2904 | 2905 | 2906 | 2907 | 2908 | 2909 | 2910 | 2911 |
| 5540 | 2912 | 2913 | 2914 | 2915 | 2916 | 2917 | 2918 | 2919 |
| 5550 | 2920 | 2921 | 2922 | 2923 | 2924 | 2925 | 2926 | 2927 |
| 5560 | 2928 | 2929 | 2930 | 2931 | 2932 | 2933 | 2934 | 2935 |
| 5570 | 2936 | 2937 | 2938 | 2939 | 2940 | 2941 | 2942 | 2943 |
| 5600 | 2944 | 2945 | 2946 | 2947 | 2948 | 2949 | 2950 | 2951 |
| 5610 | 2952 | 2953 | 2954 | 2955 | 2956 | 2957 | 2958 | 2959 |
| 5620 | 2960 | 2961 | 2962 | 2963 | 2964 | 2965 | 2966 | 2967 |
| 5630 | 2968 | 2969 | 2970 | 2971 | 2972 | 2973 | 2974 | 2975 |
| 5640 | 2976 | 2977 | 2978 | 2979 | 2980 | 2981 | 2982 | 2983 |
| 5650 | 2984 | 2985 | 2986 | 2987 | 2988 | 2989 | 2990 | 2991 |
| 5660 | 2992 | 2993 | 2994 | 2995 | 2996 | 2997 | 2998 | 2999 |
| 5670 | 3000 | 3001 | 3002 | 3003 | 3004 | 3005 | 3006 | 3007 |
| 5700 | 3008 | 3009 | 3010 | 3011 | 3012 | 3013 | 3014 | 3015 |
| 5710 | 3016 | 3017 | 3018 | 3019 | 3020 | 3021 | 3022 | 3023 |
| 5720 | 3024 | 3025 | 3026 | 3027 | 3028 | 3029 | 3030 | 3031 |
| 5730 | 3032 | 3033 | 3034 | 3035 | 3036 | 3037 | 3038 | 3039 |
| 5740 | 3040 | 3041 | 3042 | 3043 | 3044 | 3045 | 3046 | 3047 |
| 5750 | 3048 | 3049 | 3050 | 3051 | 3052 | 3053 | 3054 | 3055 |
| 5760 | 3056 | 3057 | 3058 | 3059 | 3060 | 3061 | 3062 | 3063 |
| 5770 | 3064 | 3065 | 3066 | 3067 | 3068 | 3069 | 3070 | 3071 |

# OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 6000 | 3072 | 3073 | 3074 | 3075 | 3076 | 3077 | 3078 | 3079 |
| 6010 | 3080 | 3081 | 3082 | 3083 | 3084 | 3085 | 3086 | 3087 |
| 6020 | 3088 | 3089 | 3090 | 3091 | 3092 | 3093 | 3094 | 3095 |
| 6030 | 3096 | 3097 | 3098 | 3099 | 3100 | 3101 | 3102 | 3103 |
| 6040 | 3104 | 3105 | 3106 | 3107 | 3108 | 3109 | 3110 | 3111 |
| 6050 | 3112 | 3113 | 3114 | 3115 | 3116 | 3117 | 3118 | 3119 |
| 6060 | 3120 | 3121 | 3122 | 3123 | 3124 | 3125 | 3126 | 3127 |
| 6070 | 3128 | 3129 | 3130 | 3131 | 3132 | 3133 | 3134 | 3135 |
| 6100 | 3136 | 3137 | 3138 | 3139 | 3140 | 3141 | 3142 | 3143 |
| 6110 | 3144 | 3145 | 3146 | 3147 | 3148 | 3149 | 3150 | 3151 |
| 6120 | 3152 | 3153 | 3154 | 3155 | 3156 | 3157 | 3158 | 3159 |
| 6130 | 3160 | 3161 | 3162 | 3163 | 3164 | 3165 | 3166 | 3167 |
| 6140 | 3168 | 3169 | 3170 | 3171 | 3172 | 3173 | 3174 | 3175 |
| 6150 | 3176 | 3177 | 3178 | 3179 | 3180 | 3181 | 3182 | 3183 |
| 6160 | 3184 | 3185 | 3186 | 3187 | 3188 | 3189 | 3190 | 3191 |
| 6170 | 3192 | 3193 | 3194 | 3195 | 3196 | 3197 | 3198 | 3199 |
| 6200 | 3200 | 3201 | 3202 | 3203 | 3204 | 3205 | 3206 | 3207 |
| 6210 | 3208 | 3209 | 3210 | 3211 | 3212 | 3213 | 3214 | 3215 |
| 6220 | 3216 | 3217 | 3218 | 3219 | 3220 | 3221 | 3222 | 3223 |
| 6230 | 3224 | 3225 | 3226 | 3227 | 3228 | 3229 | 3230 | 3231 |
| 6240 | 3232 | 3233 | 3234 | 3235 | 3236 | 3237 | 3238 | 3239 |
| 6250 | 3240 | 3241 | 3242 | 3243 | 3244 | 3245 | 3246 | 3247 |
| 6260 | 3248 | 3249 | 3250 | 3251 | 3252 | 3253 | 3254 | 3255 |
| 6270 | 3256 | 3257 | 3258 | 3259 | 3260 | 3261 | 3262 | 3263 |
| 6300 | 3264 | 3265 | 3266 | 3267 | 3268 | 3269 | 3270 | 3271 |
| 6310 | 3272 | 3273 | 3274 | 3275 | 3276 | 3277 | 3278 | 3279 |
| 6320 | 3280 | 3281 | 3282 | 3283 | 3284 | 3285 | 3286 | 3287 |
| 6330 | 3288 | 3289 | 3290 | 3291 | 3292 | 3293 | 3294 | 3295 |
| 6340 | 3296 | 3297 | 3298 | 3299 | 3300 | 3301 | 3302 | 3303 |
| 6350 | 3304 | 3305 | 3306 | 3307 | 3308 | 3309 | 3310 | 3311 |
| 6360 | 3312 | 3313 | 3314 | 3315 | 3316 | 3317 | 3318 | 3319 |
| 6370 | 3320 | 3321 | 3322 | 3323 | 3324 | 3325 | 3326 | 3327 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 6400 | 3328 | 3329 | 3330 | 3331 | 3332 | 3333 | 3334 | 3335 |
| 6410 | 3336 | 3337 | 3338 | 3339 | 3340 | 3341 | 3342 | 3343 |
| 6420 | 3344 | 3345 | 3346 | 3347 | 3348 | 3349 | 3350 | 3351 |
| 6430 | 3352 | 3353 | 3354 | 3355 | 3356 | 3357 | 3358 | 3359 |
| 6440 | 3360 | 3361 | 3362 | 3363 | 3364 | 3365 | 3366 | 3367 |
| 6450 | 3368 | 3369 | 3370 | 3371 | 3372 | 3373 | 3374 | 3375 |
| 6460 | 3376 | 3377 | 3378 | 3379 | 3380 | 3381 | 3382 | 3383 |
| 6470 | 3384 | 3385 | 3386 | 3387 | 3388 | 3389 | 3390 | 3391 |
| 6500 | 3392 | 3393 | 3394 | 3395 | 3396 | 3397 | 3398 | 3399 |
| 6510 | 3400 | 3401 | 3402 | 3403 | 3404 | 3405 | 3406 | 3407 |
| 6520 | 3408 | 3409 | 3410 | 3411 | 3412 | 3413 | 3414 | 3415 |
| 6530 | 3416 | 3417 | 3418 | 3419 | 3420 | 3421 | 3422 | 3423 |
| 6540 | 3424 | 3425 | 3426 | 3427 | 3428 | 3429 | 3430 | 3431 |
| 6550 | 3432 | 3433 | 3434 | 3435 | 3436 | 3437 | 3438 | 3439 |
| 6560 | 3440 | 3441 | 3442 | 3443 | 3444 | 3445 | 3446 | 3447 |
| 6570 | 3448 | 3449 | 3450 | 3451 | 3452 | 3453 | 3454 | 3455 |
| 6600 | 3456 | 3457 | 3458 | 3459 | 3460 | 3461 | 3462 | 3463 |
| 6610 | 3464 | 3465 | 3466 | 3467 | 3468 | 3469 | 3470 | 3471 |
| 6620 | 3472 | 3473 | 3474 | 3475 | 3476 | 3477 | 3478 | 3479 |
| 6630 | 3480 | 3481 | 3482 | 3483 | 3484 | 3485 | 3486 | 3487 |
| 6640 | 3488 | 3489 | 3490 | 3491 | 3492 | 3493 | 3494 | 3495 |
| 6650 | 3496 | 3497 | 3498 | 3499 | 3500 | 3501 | 3502 | 3503 |
| 6660 | 3504 | 3505 | 3506 | 3507 | 3508 | 3509 | 3510 | 3511 |
| 6670 | 3512 | 3513 | 3514 | 3515 | 3516 | 3517 | 3518 | 3519 |
| 6700 | 3520 | 3521 | 3522 | 3523 | 3524 | 3525 | 3526 | 3527 |
| 6710 | 3528 | 3529 | 3530 | 3531 | 3532 | 3533 | 3534 | 3535 |
| 6720 | 3536 | 3537 | 3538 | 3539 | 3540 | 3541 | 3542 | 3543 |
| 6730 | 3544 | 3545 | 3546 | 3547 | 3548 | 3549 | 3550 | 3551 |
| 6740 | 3552 | 3553 | 3554 | 3555 | 3556 | 3557 | 3558 | 3559 |
| 6750 | 3560 | 3561 | 3562 | 3563 | 3564 | 3565 | 3566 | 3567 |
| 6760 | 3568 | 3569 | 3570 | 3571 | 3572 | 3573 | 3574 | 3575 |
| 6770 | 3576 | 3577 | 3578 | 3579 | 3580 | 3581 | 3582 | 3583 |

| 6000 to 6777 (Octal) | 3072 to 3583 (Decimal) |
|---|---|

| Octal | Decimal |
|---|---|
| 10000 - | 4096 |
| 20000 - | 8192 |
| 30000 - | 12288 |
| 40000 - | 16384 |
| 50000 - | 20480 |
| 60000 - | 24576 |
| 70000 - | 28672 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 7000 | 3584 | 3585 | 3586 | 3587 | 3588 | 3589 | 3590 | 3591 |
| 7010 | 3592 | 3593 | 3594 | 3595 | 3596 | 3597 | 3598 | 3599 |
| 7020 | 3600 | 3601 | 3602 | 3603 | 3604 | 3605 | 3606 | 3607 |
| 7030 | 3608 | 3609 | 3610 | 3611 | 3612 | 3613 | 3614 | 3615 |
| 7040 | 3616 | 3617 | 3618 | 3619 | 3620 | 3621 | 3622 | 3623 |
| 7050 | 3624 | 3625 | 3626 | 3627 | 3628 | 3629 | 3630 | 3631 |
| 7060 | 3632 | 3633 | 3634 | 3635 | 3636 | 3637 | 3638 | 3639 |
| 7070 | 3640 | 3641 | 3642 | 3643 | 3644 | 3645 | 3646 | 3647 |
| 7100 | 3648 | 3649 | 3650 | 3651 | 3652 | 3653 | 3654 | 3655 |
| 7110 | 3656 | 3657 | 3658 | 3659 | 3660 | 3661 | 3662 | 3663 |
| 7120 | 3664 | 3665 | 3666 | 3667 | 3668 | 3669 | 3670 | 3671 |
| 7130 | 3672 | 3673 | 3674 | 3675 | 3676 | 3677 | 3678 | 3679 |
| 7140 | 3680 | 3681 | 3682 | 3683 | 3684 | 3685 | 3686 | 3687 |
| 7150 | 3688 | 3689 | 3690 | 3691 | 3692 | 3693 | 3694 | 3695 |
| 7160 | 3696 | 3697 | 3698 | 3699 | 3700 | 3701 | 3702 | 3703 |
| 7170 | 3704 | 3705 | 3706 | 3707 | 3708 | 3709 | 3710 | 3711 |
| 7200 | 3712 | 3713 | 3714 | 3715 | 3716 | 3717 | 3718 | 3719 |
| 7210 | 3720 | 3721 | 3722 | 3723 | 3724 | 3725 | 3726 | 3727 |
| 7220 | 3728 | 3729 | 3730 | 3731 | 3732 | 3733 | 3734 | 3735 |
| 7230 | 3736 | 3737 | 3738 | 3739 | 3740 | 3741 | 3742 | 3743 |
| 7240 | 3744 | 3745 | 3746 | 3747 | 3748 | 3749 | 3750 | 3751 |
| 7250 | 3752 | 3753 | 3754 | 3755 | 3756 | 3757 | 3758 | 3759 |
| 7260 | 3760 | 3761 | 3762 | 3763 | 3764 | 3765 | 3766 | 3767 |
| 7270 | 3768 | 3769 | 3770 | 3771 | 3772 | 3773 | 3774 | 3775 |
| 7300 | 3776 | 3777 | 3778 | 3779 | 3780 | 3781 | 3782 | 3783 |
| 7310 | 3784 | 3785 | 3786 | 3787 | 3788 | 3789 | 3790 | 3791 |
| 7320 | 3792 | 3793 | 3794 | 3795 | 3796 | 3797 | 3798 | 3799 |
| 7330 | 3800 | 3801 | 3802 | 3803 | 3804 | 3805 | 3806 | 3807 |
| 7340 | 3808 | 3809 | 3810 | 3811 | 3812 | 3813 | 3814 | 3815 |
| 7350 | 3816 | 3817 | 3818 | 3819 | 3820 | 3821 | 3822 | 3823 |
| 7360 | 3824 | 3825 | 3826 | 3827 | 3828 | 3829 | 3830 | 3831 |
| 7370 | 3832 | 3833 | 3834 | 3835 | 3836 | 3837 | 3838 | 3839 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 7400 | 3840 | 3841 | 3842 | 3843 | 3844 | 3845 | 3846 | 3847 |
| 7410 | 3848 | 3849 | 3850 | 3851 | 3852 | 3853 | 3854 | 3855 |
| 7420 | 3856 | 3857 | 3858 | 3859 | 3860 | 3861 | 3862 | 3863 |
| 7430 | 3864 | 3865 | 3866 | 3867 | 3868 | 3869 | 3870 | 3871 |
| 7440 | 3872 | 3873 | 3874 | 3875 | 3876 | 3877 | 3878 | 3879 |
| 7450 | 3880 | 3881 | 3882 | 3883 | 3884 | 3885 | 3886 | 3887 |
| 7460 | 3888 | 3889 | 3890 | 3891 | 3892 | 3893 | 3894 | 3895 |
| 7470 | 3896 | 3897 | 3898 | 3899 | 3900 | 3901 | 3902 | 3903 |
| 7500 | 3904 | 3905 | 3906 | 3907 | 3908 | 3909 | 3910 | 3911 |
| 7510 | 3912 | 3913 | 3914 | 3915 | 3916 | 3917 | 3918 | 3919 |
| 7520 | 3920 | 3921 | 3922 | 3923 | 3924 | 3925 | 3926 | 3927 |
| 7530 | 3928 | 3929 | 3930 | 3931 | 3932 | 3933 | 3934 | 3935 |
| 7540 | 3936 | 3937 | 3938 | 3939 | 3940 | 3941 | 3942 | 3943 |
| 7550 | 3944 | 3945 | 3946 | 3947 | 3948 | 3949 | 3950 | 3951 |
| 7560 | 3952 | 3953 | 3954 | 3955 | 3956 | 3957 | 3958 | 3959 |
| 7570 | 3960 | 3961 | 3962 | 3963 | 3964 | 3965 | 3966 | 3967 |
| 7600 | 3968 | 3969 | 3970 | 3971 | 3972 | 3973 | 3974 | 3975 |
| 7610 | 3976 | 3977 | 3978 | 3979 | 3980 | 3981 | 3982 | 3983 |
| 7620 | 3984 | 3985 | 3986 | 3987 | 3988 | 3989 | 3990 | 3991 |
| 7630 | 3992 | 3993 | 3994 | 3995 | 3996 | 3997 | 3998 | 3999 |
| 7640 | 4000 | 4001 | 4002 | 4003 | 4004 | 4005 | 4006 | 4007 |
| 7650 | 4008 | 4009 | 4010 | 4011 | 4012 | 4013 | 4014 | 4015 |
| 7660 | 4016 | 4017 | 4018 | 4019 | 4020 | 4021 | 1022 | 4023 |
| 7670 | 4024 | 4025 | 4026 | 4027 | 4028 | 4029 | 4030 | 4031 |
| 7700 | 4032 | 4033 | 4034 | 4035 | 4036 | 4037 | 4038 | 4039 |
| 7710 | 4040 | 4041 | 4042 | 4043 | 4044 | 4045 | 4046 | 4047 |
| 7720 | 4048 | 4049 | 4050 | 4051 | 4052 | 4053 | 4054 | 4055 |
| 7730 | 4056 | 4057 | 4058 | 4059 | 4060 | 4061 | 4062 | 4063 |
| 7740 | 4064 | 4065 | 4066 | 4067 | 4068 | 4069 | 4070 | 4071 |
| 7750 | 4072 | 4073 | 4074 | 4075 | 4076 | 4077 | 4078 | 4079 |
| 7760 | 4080 | 4081 | 4082 | 4083 | 4084 | 4085 | 4086 | 4087 |
| 7770 | 4088 | 4089 | 4090 | 4091 | 4092 | 4093 | 4094 | 4095 |

| 7000 to 7777 (Octal) | 3584 to 4095 (Decimal) |
|---|---|

# OCTAL-DECIMAL FRACTION CONVERSION TABLE

| Octal | Decimal | Octal | Decimal | Octal | Decimal | Octal | Decimal |
|---|---|---|---|---|---|---|---|
| .000 | .000000 | .100 | .125000 | .200 | .250000 | .300 | .375000 |
| .001 | .001953 | .101 | .126953 | .201 | .251953 | .301 | .376953 |
| .002 | .003906 | .102 | .128906 | .202 | .253906 | .302 | .378906 |
| .003 | .005859 | .103 | .130859 | .203 | .255859 | .303 | .380859 |
| .004 | .007812 | .104 | .132812 | .204 | .257812 | .304 | .382812 |
| .005 | .009765 | .105 | .134765 | .205 | .259765 | .305 | .384765 |
| .006 | .011718 | .106 | .136718 | .206 | .261718 | .306 | .386718 |
| .007 | .013671 | .107 | .138671 | .207 | .263671 | .307 | .388671 |
| .010 | .015625 | .110 | .140625 | .210 | .265625 | .310 | .390625 |
| .011 | .017578 | .111 | .142578 | .211 | .267578 | .311 | .392578 |
| .012 | .019531 | .112 | .144531 | .212 | .269531 | .312 | .394531 |
| .013 | .021484 | .113 | .146484 | .213 | .271484 | .313 | .396484 |
| .014 | .023437 | .114 | .148437 | .214 | .273437 | .314 | .398437 |
| .015 | .025390 | .115 | .150390 | .215 | .275390 | .315 | .400390 |
| .016 | .027343 | .116 | .152343 | .216 | .277343 | .316 | .402343 |
| .017 | .029296 | .117 | .154296 | .217 | .279296 | .317 | .404296 |
| .020 | .031250 | .120 | .156250 | .220 | .281250 | .320 | .406250 |
| .021 | .033203 | .121 | .158203 | .221 | .283203 | .321 | .408203 |
| .022 | .035156 | .122 | .160156 | .222 | .285156 | .322 | .410156 |
| .023 | .037109 | .123 | .162109 | .223 | .287109 | .323 | .412109 |
| .024 | .039062 | .124 | .164062 | .224 | .289062 | .324 | .414062 |
| .025 | .041015 | .125 | .166015 | .225 | .291015 | .325 | .416015 |
| .026 | .042968 | .126 | .167968 | .226 | .292968 | .326 | .417968 |
| .027 | .044921 | .127 | .169921 | .227 | .294921 | .327 | .419921 |
| .030 | .046875 | .130 | .171875 | .230 | .296875 | .330 | .421875 |
| .031 | .048828 | .131 | .173828 | .231 | .298828 | .331 | .423828 |
| .032 | .050781 | .132 | .175781 | .232 | .300781 | .332 | .425781 |
| .033 | .052734 | .133 | .177734 | .233 | .302734 | .333 | .427734 |
| .034 | .054687 | .134 | .179687 | .234 | .304687 | .334 | .429687 |
| .035 | .056640 | .135 | .181640 | .235 | .306640 | .335 | .431640 |
| .036 | .058593 | .136 | .183593 | .236 | .308593 | .336 | .433593 |
| .037 | .060546 | .137 | .185546 | .237 | .310546 | .337 | .435546 |
| .040 | .062500 | .140 | .187500 | .240 | .312500 | .340 | .437500 |
| .041 | .064453 | .141 | .189453 | .241 | .314453 | .341 | .439453 |
| .042 | .066406 | .142 | .191406 | .242 | .316406 | .342 | .441406 |
| .043 | .068359 | .143 | .193359 | .243 | .318359 | .343 | .443359 |
| .044 | .070312 | .144 | .195312 | .244 | .320312 | .344 | .445312 |
| .045 | .072265 | .145 | .197265 | .245 | .322265 | .345 | .447265 |
| .046 | .074218 | .146 | .199218 | .246 | .324218 | .346 | .449218 |
| .047 | .076171 | .147 | .201171 | .247 | .326171 | .347 | .451171 |
| .050 | .078125 | .150 | .203125 | .250 | .328125 | .350 | .453125 |
| .051 | .080078 | .151 | .205078 | .251 | .330078 | .351 | .455078 |
| .052 | .082031 | .152 | .207031 | .252 | .332031 | .352 | .457031 |
| .053 | .083984 | .153 | .208984 | .253 | .333984 | .353 | .458984 |
| .054 | .085937 | .154 | .210937 | .254 | .335937 | .354 | .460937 |
| .055 | .087890 | .155 | .212890 | .255 | .337890 | .355 | .462890 |
| .056 | .089843 | .156 | .214843 | .256 | .339843 | .356 | .464843 |
| .057 | .091796 | .157 | .216796 | .257 | .341796 | .357 | .466796 |
| .060 | .093750 | .160 | .218750 | .260 | .343750 | .360 | .468750 |
| .061 | .095703 | .161 | .220703 | .261 | .345703 | .361 | .470703 |
| .062 | .097656 | .162 | .222656 | .262 | .347656 | .362 | .472656 |
| .063 | .099609 | .163 | .224609 | .263 | .349609 | .363 | .474609 |
| .064 | .101562 | .164 | .226562 | .264 | .351562 | .364 | .476562 |
| .065 | .103515 | .165 | .228515 | .265 | .353515 | .365 | .478515 |
| .066 | .105468 | .166 | .230468 | .266 | .355468 | .366 | .480468 |
| .067 | .107421 | .167 | .232421 | .267 | .357421 | .367 | .482421 |
| .070 | .109375 | .170 | .234375 | .270 | .359375 | .370 | .484375 |
| .071 | .111328 | .171 | .236328 | .271 | .361328 | .371 | .486328 |
| .072 | .113281 | .172 | .238281 | .272 | .363281 | .372 | .488281 |
| .073 | .115234 | .173 | .240234 | .273 | .365234 | .373 | .490234 |
| .074 | .117187 | .174 | .242187 | .274 | .367187 | .374 | .492187 |
| .075 | .119140 | .175 | .244140 | .275 | .369140 | .375 | .494140 |
| .076 | .121093 | .176 | .246093 | .276 | .371093 | .376 | .496093 |
| .077 | .123046 | .177 | .248046 | .277 | .373046 | .377 | .498046 |

# OCTAL-DECIMAL FRACTION CONVERSION TABLE (continued)

| Octal | Decimal | Octal | Decimal | Octal | Decimal | Octal | Decimal |
|---|---|---|---|---|---|---|---|
| .000000 | .000000 | .000100 | .000244 | .000200 | .000488 | .000300 | .000732 |
| .000001 | .000003 | .000101 | .000247 | .000201 | .000492 | .000301 | .000736 |
| .000002 | .000007 | .000102 | .000251 | .000202 | .000495 | .000302 | .000740 |
| .000003 | .000011 | .000103 | .000255 | .000203 | .000499 | .000303 | .000743 |
| .000004 | .000015 | .000104 | .000259 | .000204 | .000503 | .000304 | .000747 |
| .000005 | .000019 | .000105 | .000263 | .000205 | .000507 | .000305 | .000751 |
| .000006 | .000022 | .000106 | .000267 | .000206 | .000511 | .000306 | .000755 |
| .000007 | .000026 | .000107 | .000270 | .000207 | .000514 | .000307 | .000759 |
| .000010 | .000030 | .000110 | .000274 | .000210 | .000518 | .000310 | .000762 |
| .000011 | .000034 | .000111 | .000278 | .000211 | .000522 | .000311 | .000766 |
| .000012 | .000038 | .000112 | .000282 | .000212 | .000526 | .000312 | .000770 |
| .000013 | .000041 | .000113 | .000286 | .000213 | .000530 | .000313 | .000774 |
| .000014 | .000045 | .000114 | .000289 | .000214 | .000534 | .000314 | .000778 |
| .000015 | .000049 | .000115 | .000293 | .000215 | .000537 | .000315 | .000782 |
| .000016 | .000053 | .000116 | .000297 | .000216 | .000541 | .000316 | .000785 |
| .000017 | .000057 | .000117 | .000301 | .000217 | .000545 | .000317 | .000789 |
| .000020 | .000061 | .000120 | .000305 | .000220 | .000549 | .000320 | .000793 |
| .000021 | .000064 | .000121 | .000308 | .000221 | .000553 | .000321 | .000797 |
| .000022 | .000068 | .000122 | .000312 | .000222 | .000556 | .000322 | .000801 |
| .000023 | .000072 | .000123 | .000316 | .000223 | .000560 | .000323 | .000805 |
| .000024 | .000076 | .000124 | .000320 | .000224 | .000564 | .000324 | .000808 |
| .000025 | .000080 | .000125 | .000324 | .000225 | .000568 | .000325 | .000812 |
| .000026 | .000083 | .000126 | .000328 | .000226 | .000572 | .000326 | .000816 |
| .000027 | .000087 | .000127 | .000331 | .000227 | .000576 | .000327 | .000820 |
| .000030 | .000091 | .000130 | .000335 | .000230 | .000579 | .000330 | .000823 |
| .000031 | .000095 | .000131 | .000339 | .000231 | .000583 | .000331 | .000827 |
| .000032 | .000099 | .000132 | .000343 | .000232 | .000587 | .000332 | .000831 |
| .000033 | .000102 | .000133 | .000347 | .000233 | .000591 | .000333 | .000835 |
| .000034 | .000106 | .000134 | .000350 | .000234 | .000595 | .000334 | .000839 |
| .000035 | .000110 | .000135 | .000354 | .000235 | .000598 | .000335 | .000843 |
| .000036 | .000114 | .000136 | .000358 | .000236 | .000602 | .000336 | .000846 |
| .000037 | .000118 | .000137 | .000362 | .000237 | .000606 | .000337 | .000850 |
| .000040 | .000122 | .000140 | .000366 | .000240 | .000610 | .000340 | .000854 |
| .000041 | .000125 | .000141 | .000370 | .000241 | .000614 | .000341 | .000858 |
| .000042 | .000129 | .000142 | .000373 | .000242 | .000617 | .000342 | .000862 |
| .000043 | .000133 | .000143 | .000377 | .000243 | .000621 | .000343 | .000865 |
| .000044 | .000137 | .000144 | .000381 | .000244 | .000625 | .000344 | .000869 |
| .000045 | .000141 | .000145 | .000385 | .000245 | .000629 | .000345 | .000873 |
| .000046 | .000144 | .000146 | .000389 | .000246 | .000633 | .000346 | .000877 |
| .000047 | .000148 | .000147 | .000392 | .000247 | .000637 | .000347 | .000881 |
| .000050 | .000152 | .000150 | .000396 | .000250 | .000640 | .000350 | .000885 |
| .000051 | .000156 | .000151 | .000400 | .000251 | .000644 | .000351 | .000888 |
| .000052 | .000160 | .000152 | .000404 | .000252 | .000648 | .000352 | .000892 |
| .000053 | .000164 | .000153 | .000408 | .000253 | .000652 | .000353 | .000896 |
| .000054 | .000167 | .000154 | .000411 | .000254 | .000656 | .000354 | .000900 |
| .000055 | .000171 | .000155 | .000415 | .000255 | .000659 | .000355 | .000904 |
| .000056 | .000175 | .000156 | .000419 | .000256 | .000663 | .000356 | .000907 |
| .000057 | .000179 | .000157 | .000423 | .000257 | .000667 | .000357 | .000911 |
| .000060 | .000183 | .000160 | .000427 | .000260 | .000671 | .000360 | .000915 |
| .000061 | .000186 | .000161 | .000431 | .000261 | .000675 | .000361 | .000919 |
| .000062 | .000190 | .000162 | .000434 | .000262 | .000679 | .000362 | .000923 |
| .000063 | .000194 | .000163 | .000438 | .000263 | .000682 | .000363 | .000926 |
| .000064 | .000198 | .000164 | .000442 | .000264 | .000686 | .000364 | .000930 |
| .000065 | .000202 | .000165 | .000446 | .000265 | .000690 | .000365 | .000934 |
| .000066 | .000205 | .000166 | .000450 | .000266 | .000694 | .000366 | .000938 |
| .000067 | .000209 | .000167 | .000453 | .000267 | .000698 | .000367 | .000942 |
| .000070 | .000213 | .000170 | .000457 | .000270 | .000701 | .000370 | .000946 |
| .000071 | .000217 | .000171 | .000461 | .000271 | .000705 | .000371 | .000949 |
| .000072 | .000221 | .000172 | .000465 | .000272 | .000709 | .000372 | .000953 |
| .000073 | .000225 | .000173 | .000469 | .000273 | .000713 | .000373 | .000957 |
| .000074 | .000228 | .000174 | .000473 | .000274 | .000717 | .000374 | .000961 |
| .000075 | .000232 | .000175 | .000476 | .000275 | .000720 | .000375 | .000965 |
| .000076 | .000236 | .000176 | .000480 | .000276 | .000724 | .000376 | .000968 |
| .000077 | .000240 | .000177 | .000484 | .000277 | .000728 | .000377 | .000972 |

# OCTAL-DECIMAL FRACTION CONVERSION TABLE (continued)

| Octal | Decimal | Octal | Decimal | Octal | Decimal | Octal | Decimal |
|---|---|---|---|---|---|---|---|
| .000400 | .000976 | .000500 | .001220 | .000600 | .001464 | .000700 | .001708 |
| .000401 | .000980 | .000501 | .001224 | .000601 | .001468 | .000701 | .001712 |
| .000402 | .000984 | .000502 | .001228 | .000602 | .001472 | .000702 | .001716 |
| .000403 | .000988 | .000503 | .001232 | .000603 | .001476 | .000703 | .001720 |
| .000404 | .000991 | .000504 | .001235 | .000604 | .001480 | .000704 | .001724 |
| .000405 | .000995 | .000505 | .001239 | .000605 | .001483 | .000705 | .001728 |
| .000406 | .000999 | .000506 | .001243 | .000606 | .001487 | .000706 | .001731 |
| .000407 | .001003 | .000507 | .001247 | .000607 | .001491 | .000707 | .001735 |
| .000410 | .001007 | .000510 | .001251 | .000610 | .001495 | .000710 | .001739 |
| .000411 | .001010 | .000511 | .001255 | .000611 | .001499 | .000711 | .001743 |
| .000412 | .001014 | .000512 | .001258 | .000612 | .001502 | .000712 | .001747 |
| .000413 | .001018 | .000513 | .001262 | .000613 | .001506 | .000713 | .001750 |
| .000414 | .001022 | .000514 | .001266 | .000614 | .001510 | .000714 | .001754 |
| .000415 | .001026 | .000515 | .001270 | .000615 | .001514 | .000715 | .001758 |
| .000416 | .001029 | .000516 | .001274 | .000616 | .001518 | .000716 | .001762 |
| .000417 | .001033 | .000517 | .001277 | .000617 | .001522 | .000717 | .001766 |
| .000420 | .001037 | .000520 | .001281 | .000620 | .001525 | .000720 | .001770 |
| .000421 | .001041 | .000521 | .001285 | .000621 | .001529 | .000721 | .001773 |
| .000422 | .001045 | .000522 | .001289 | .000622 | .001533 | .000722 | .001777 |
| .000423 | .001049 | .000523 | .001293 | .000623 | .001537 | .000723 | .001781 |
| .000424 | .001052 | .000524 | .001296 | .000624 | .001541 | .000724 | .001785 |
| .000425 | .001056 | .000525 | .001300 | .000625 | .001544 | .000725 | .001789 |
| .000426 | .001060 | .000526 | .001304 | .000626 | .001548 | .000726 | .001792 |
| .000427 | .001064 | .000527 | .001308 | .000627 | .001552 | .000727 | .001796 |
| .000430 | .001068 | .000530 | .001312 | .000630 | .001556 | .000730 | .001800 |
| .000431 | .001071 | .000531 | .001316 | .000631 | .001560 | .000731 | .001804 |
| .000432 | .001075 | .000532 | .001319 | .000632 | .001564 | .000732 | .001808 |
| .000433 | .001079 | .000533 | .001323 | .000633 | .001567 | .000733 | .001811 |
| .000434 | .001083 | .000534 | .001327 | .000634 | .001571 | .000734 | .001815 |
| .000435 | .001087 | .000535 | .001331 | .000635 | .001575 | .000735 | .001819 |
| .000436 | .001091 | .000536 | .001335 | .000636 | .001579 | .000736 | .001823 |
| .000437 | .001094 | .000537 | .001338 | .000637 | .001583 | .000737 | .001827 |
| .000440 | .001098 | .000540 | .001342 | .000640 | .001586 | .000740 | .001831 |
| .000441 | .001102 | .000541 | .001346 | .000641 | .001590 | .000741 | .001834 |
| .000442 | .001106 | .000542 | .001350 | .000642 | .001594 | .000742 | .001838 |
| .000443 | .001110 | .000543 | .001354 | .000643 | .001598 | .000743 | .001842 |
| .000444 | .001113 | .000544 | .001358 | .000644 | .001602 | .000744 | .001846 |
| .000445 | .001117 | .000545 | .001361 | .000645 | .001605 | .000745 | .001850 |
| .000446 | .001121 | .000546 | .001365 | .000646 | .001609 | .000746 | .001853 |
| .000447 | .001125 | .000547 | .001369 | .000647 | .001613 | .000747 | .001857 |
| .000450 | .001129 | .000550 | .001373 | .000650 | .001617 | .000750 | .001861 |
| .000451 | .001132 | .000551 | .001377 | .000651 | .001621 | .000751 | .001865 |
| .000452 | .001136 | .000552 | .001380 | .000652 | .001625 | .000752 | .001869 |
| .000453 | .001140 | .000553 | .001384 | .000653 | .001628 | .000753 | .001873 |
| .000454 | .001144 | .000554 | .001388 | .000654 | .001632 | .000754 | .001876 |
| .000455 | .001148 | .000555 | .001392 | .000655 | .001636 | .000755 | .001880 |
| .000456 | .001152 | .000556 | .001396 | .000656 | .001640 | .000756 | .001884 |
| .000457 | .001155 | .000557 | .001399 | .000657 | .001644 | .000757 | .001888 |
| .000460 | .001159 | .000560 | .001403 | .000660 | .001647 | .000760 | .001892 |
| .000461 | .001163 | .000561 | .001407 | .000661 | .001651 | .000761 | .001895 |
| .000462 | .001167 | .000562 | .001411 | .000662 | .001655 | .000762 | .001899 |
| .000463 | .001171 | .000563 | .001415 | .000663 | .001659 | .000763 | .001903 |
| .000464 | .001174 | .000564 | .001419 | 000664 | .001663 | .000764 | .001907 |
| .000465 | .001178 | .000565 | .001422 | .000665 | .001667 | .000765 | .001911 |
| .000466 | .001182 | .000566 | .001426 | .000666 | .001670 | .000766 | .001914 |
| .000467 | .001186 | .000567 | .001430 | .000667 | .001674 | .000767 | .001918 |
| .000470 | .001190 | .000570 | .001434 | .000670 | .001678 | .000770 | .001922 |
| .000471 | .001194 | .000571 | .001438 | .000671 | .001682 | .000771 | .001926 |
| .000472 | .001197 | .000572 | .001441 | .000672 | .001686 | .000772 | .001930 |
| .000473 | .001201 | .000573 | .001445 | .000673 | .001689 | .000773 | .001934 |
| .000474 | .001205 | .000574 | .001449 | .000674 | .001693 | .000774 | .001937 |
| .000475 | .001209 | .000575 | .001453 | .000675 | .001697 | .000775 | .001941 |
| .000476 | .001213 | .000576 | .001457 | .000676 | .001701 | .000776 | .001945 |
| .000477 | .001216 | .000577 | .001461 | .000677 | .001705 | .000777 | .001949 |

# APPENDIX 6
# PERFORATED-TAPE LOADER SEQUENCES

## READIN MODE LOADER

The readin mode (RIM) loader is a minimum length, basic, perforated-tape reader program for the 33 ASR. It is initially stored in memory by manual use of the operator console keys and switches. The loader is permanently stored in 18 locations of page 37.

A perforated tape to be read by the RIM loader must be in RIM format:

| Tape Channel 8 7 6 5 4 S 3 2 1 | Format |
|---|---|
| 1 0 0 0 0 . 0 0 0 | Leader-trailer code |
| 0 1 A1 . A2<br>0 0 A3 . A4 | Absolute address to contain next 4 digits |
| 0 0 X1 . X2<br>0 0 X3 . X4 | Content of previous 4-digit address |
| 0 1 A1 . A2<br>0 0 A3 . A4 | Address |
| 0 0 X1 . X2<br>0 0 X3 . X4 | Content |
| (Etc.) | (Etc.) |
| 1 0 0 0 0 . 0 0 0 | Leader-trailer code |

The RIM loader can only be used in conjunction with the 33 ASR reader (not the high-speed perforated-tape reader). Because a tape in RIM format is, in effect, twice as long as it need be, it is suggested that the RIM loader be used only to read the binary loader when using the 33 ASR. (Note that PDP-8 diagnostic program tapes are in RIM format.)

The complete PDP-8 RIM loader (SA = 7756) is as follows:

| Absolute Address | Octal Content | Tag | Instruction I Z | Comments |
|---|---|---|---|---|
| 7756, | 6032 | BEG, | KCC | /CLEAR AC AND FLAG |
| 7757, | 6031 | | KSF | /SKIP IF FLAG = 1 |
| 7760, | 5357 | | JMP .−1 | /LOOKING FOR CHARACTER |
| 7761, | 6036 | | KRB | /READ BUFFER |
| 7762, | 7106 | | CLL RTL | |
| 7763, | 7006 | | RTL | /CHANNEL 8 IN AC0 |
| 7764, | 7510 | | SPA | /CHECKING FOR LEADER |
| 7765, | 5357 | | JMP BEG+1 | /FOUND LEADER |
| 7766, | 7006 | | RTL | /OK, CHANNEL 7 IN LINK |
| 7767, | 6031 | | KSF | |
| 7770, | 5367 | | JMP .−1 | |
| 7771, | 6034 | | KRS | |
| 7772, | 7420 | | SNL | /READ, DO NOT CLEAR |
| 7773, | 3776 | | DCA I TEMP | /CHECKING FOR ADDRESS |
| 7774, | 3376 | | DCA TEMP | /STORE CONTENT |
| 7775, | 5356 | | JMP BEG | /STORE ADDRESS |
| 7776, | 0 | TEMP | 0 | /NEXT WORD |
| 7777, | JMP START OF BIN LOADER | | 0 | /TEMP STORAGE |

Placing the RIM loader in core memory by way of the operator console keys and switches is accomplished as follows:

1. Set the starting address 7756 in the switch register (SR).
2. Press LOAD ADDRESS key.
3. Set the first instruction (6032) in the SR.
4. Press the DEPOSIT key.
5. Set the next instruction (6031) in the SR.
6. Press DEPOSIT key.
7. Repeat steps 5 and 6 until all 16 instructions have been deposited.

To load a tape in RIM format, place the tape in the reader, set the SR to the starting address 7756 of the RIM loader (not of the program being read), press the LOAD ADDRESS key, press the START key, and start the Teletype reader.

Refer to Digital Program Library document Digital-8-1-U for additional information on the Readin Mode Loader program.

## BINARY LOADER

The binary loader (BIN) is used to read machine language tapes (in binary format) produced by the program assembly language (PAL). A tape in binary format is about one half the length of the comparable RIM format tape. It can, therefore, be read about twice as fast as a RIM tape and is, for this reason, the more desirable format to use with the 10 cps 33 ASR reader or the Type 750C High Speed Perforated Tape Reader.

The format of a binary tape is as follows:

LEADER: about 2 feet of leader-trailer codes.

BODY: characters representing the absolute, machine language program in easy-to-read binary (or octal) form. The section of tape may contain characters representing instructions (channels 8 and 7 not punched) or origin resettings (channel 8 not punched, channel 7 punched) and is concluded by 2 characters (channels 8 and 7 not punched) that represent a checksum for the entire section.

TRAILER: same as leader

Example of the format of a binary tape:

| Tape Channel 8 7 6 5 4 S 3 2 1 | Memory Location | Content | Comments |
|---|---|---|---|
| 1 0 0 0 0 . 0 0 0 | | | leader-trailer code |
| 0 1 0 0 0 . 0 1 0 | | | |
| 0 0 0 0 0 . 0 0 0 | | 0200 | |
| 0 0 1 1 1 . 0 1 0 | | | |
| 0 0 0 0 0 . 0 0 0 | 0200 | CLA | origin setting |
| 0 0 0 0 1 . 0 1 0 | | | |
| 0 0 1 1 1 . 1 1 1 | 0201 | TAD 277 | |
| 0 0 0 1 1 . 0 1 0 | | | |
| 0 0 1 1 1 . 1 1 0 | 0202 | DCA 276 | |
| 0 0 1 1 1 . 1 0 0 | | | |
| 0 0 0 0 0 . 0 1 0 | 0203 | HLT | |
| 0 1 0 0 0 . 0 1 0 | | | |
| 0 0 1 1 1 . 1 1 1 | | 0277 | origin setting |
| 0 0 0 0 0 . 0 0 0 | | | |
| 0 0 1 0 1 . 0 1 1 | 0277 | 0053 | |
| 0 0 0 0 1 . 0 0 0 | | | |
| 0 0 0 0 0 . 1 1 1 | | 1007 | sum check |
| 1 0 0 0 0 . 0 0 0 | | | leader-trailer code |

After a BIN tape has been read in, one of the two following conditions exists:

a. No checksum error: halt with AC = 0
b. Checksum error: halt with AC = (computed checksum) — (tape checksum)

Operation of the BIN loader in no way depends upon or uses the RIM loader. To load a tape in BIN format place the tape in the reader, set the SR to 7777 (the starting address of the BIN loader), press the LOAD ADDRESS key, set SR switch 0 up for loading via the Teletype unit or down for loading via the high speed reader, then press the START key, and start the tape reader.

Refer to Digital Program Library document Digital-8-2-U-RIM for additional information on the Binary Loader program.

# APPENDIX 7
# PROGRAMMING SYSTEM
## FEATURED PROGRAMS

The programming system for the PDP-8 consists of the MACRO-8 Symbolic Assembler, FORTRAN System compiler, Symbolic On-Line Debugging Program, Symbolic Tape Editor, Floating Point Package, mathematical function subroutines, and utility and maintenance programs. All operate with the basic computer. The programming system was designed to simplify and accelerate the process of learning to program. At the same time, experienced programmers will find that it incorporates many advanced features. The system is intended to make immediately available to each user the full, general-purpose data processing capability of the PDP-8 and to serve as the operating nucleus for a growing library of programs and routines to be made available to all installations. New techniques, routines, and programs are constantly being developed, field-tested, and documented in the Digital Program Library for incorporation in users' systems.

### MACRO-8 Symbolic Assembler

The use of an assembly program has become standard practice in programming digital computers. This process allows the programmer to code his instructions in a symbolic language, one he can work with more conveniently than the 12-bit binary numbers which actually operate the computer. The assembly program then translates the symbolic language program into its machine code equivalent. The advantages are significant: the symbolic language is more meaningful and convenient to a programmer than a numeric code; instructions or data can be referred to by symbolic names without concern for, or even knowledge of, their actual addresses in core memory; decimal and alphabetical data can be expressed in a form more convenient than binary numbers; programs can be altered without extensive changes; and debugging is considerably simplified.

The MACRO-8 Symbolic Assembler accepts source programs written in the symbolic language and converts core memory locations, computer instructions, and operand addresses from the symbolic to the binary form. It produces an object program tape, a symbol table defining memory allocations, and useful diagnostic messages.

### FORTRAN System Compiler

The FORTRAN (for FORmula TRANslation) System compiler for the PDP-8 lets the user express the problem he is trying to solve in a mixture of English words and mathematical statements that is close to the language of mathematics and is also intelligible to the computer. In addition to reducing the time needed for program preparation, the compiler enables users with little or no knowledge of the computer's organization and operating language to write effective programs for it. The FORTRAN Compiler contains the instructions the computer requires to perform the clerical work of translating the FORTRAN version of the problem statement into an object program in machine language. It also produces diagnostic messages. After compilation, the object program, the operating system and the data it will work with, are loaded into the computer for solution of the problem.

The FORTRAN language consists of four general types of statements: arithmetic, logic, control, and input/output. FORTRAN functions include addition, subtraction, multiplication, division, sine, cosine, arctangent, square root, natural logarithm, and exponential.

## Symbolic On-Line Debugging Program

On-line debugging with DDT-8 gives the user dynamic printed program status information. It gives him close control over program execution, preventing errors ("bugs") from destroying other portions of his program. He can monitor the execution of single instructions or subsections, change instructions or data in any format, and output a corrected program at the end of the debugging session.

Using the standard Teletype keyboard/reader and teleprinter/punch, the user can communicate conveniently with the PDP-8 in the symbols of his source language. He can control the execution of any portion of his object program by inserting breaks, or traps, in it. When the computer reaches a break, it transfers control of the object program to DDT. The user can then examine and modify the content of individual core memory registers to correct and improve his object program.

## Symbolic Tape Editor

The Symbolic Tape Editor program is used to edit, correct, and update symbolic program tapes using the PDP-8 and the Teletype unit. With the editor in core memory, the user reads in portions of his symbolic tape, removes, changes, or adds instructions or operands, and gets back a complete new symbolic tape with errors removed. He can work through the program instruction by instruction, spot-check it, or concentrate on new sections.

## Floating Point Package

The Floating Point Package permits the PDP-8 to perform arithmetic operations that many other computers can perform only after the addition of costly optional hardware. Floating point operations automatically align the binary points of operands, retaining the maximum precision available by discarding leading zeros. In addition to increasing accuracy, floating point operations relieve the programmer of scaling problems common in fixed point operations. This is of particular advantage to the inexperienced programmer.

## Mathematical Function Routines

The programming system also includes a set of mathematical function routines to perform the following operations in both single and double precision: addition, subtraction, multiplication, division, square root, sine, cosine, arctangent, natural logarithm, and exponential.

## Utility and Maintenance Programs

PDP-8 utility programs provide printouts or punchouts of core memory content in octal, decimal, or binary form, as specified by the user. Subroutines are provided for octal or decimal data transfer and binary-to-decimal, decimal-to-binary, and Teletype tape conversion.

A complete set of standard diagnostic programs is provided to simplify and expedite system maintenance. Program descriptions and manuals permit the user to effectively test the operation of the computer for proper core memory functioning and proper execution of instructions. In addition, diagnostic programs to check the performance of standard and optional peripheral devices are provided with the devices.

# ABSTRACTS OF PROGRAMS

The PDP-8 is delivered to the user complete with an extensive selection of system programs and routines making the full data processing capability of the new computer immediately available to each user, eliminating many commonly experienced initial programming delays.

The programs described in these abstracts come from two sources, past programming effort on the PDP-5 computer, and present and continuing programming effort on the PDP-8. Thus the PDP-8 programming system takes advantage of the many man-years of program development and field testing by PDP-5 users.

Although in many cases PDP-8 programs originated as PDP-5 programs, all utility and functional program documentation is issued in a new, recursive format introduced with the PDP-8. Programs written by users of either the PDP-5 or the PDP-8 and submitted to the DECUS library (DECUS - Digital Equipment Corporation Users' Society) are immediately available to PDP-8 users. Consequently, users of either computer can take advantage of the continuing program developments for the other.

## System Programs

Digital-8-1-S

Symbolic Editor

The Symbolic Editor program is used to generate, edit, correct, and update symbolic program tapes using the tape teleprinter. With the Editor in memory, the user reads in portions of his symbolic tape, removes, changes, or adds instructions or operands, and gets back a new, complete, symbolic tape with errors removed. He can work through the program instruction by instruction, spot check it, or concentrate on new sections. The tape can contain either symbolic machine language, FORTRAN source statement, data, or text information. This program is available for use with either the 33ASR reader/punch or the high speed reader/punch.

Digital-8-2-S

FORTRAN System

One-pass FORTRAN compiler and operating system compiles FORTRAN source language statements into an object program tape. The operating system executes the program. This system contains the interpreter, arithmetic function subroutines, and input/output packages.

Digital-8-3-S

PAL III (Program Assembler Language)

Symbolic machine language assembler. Converts programs coded in symbolic machine language to binary machine language. The basic process performed by the Assembler is the substitution

of numeric values for symbols, according to associations defined in the symbol table. In addition, the user may request that the Assembler itself assign values to the user's own symbols at assembly time. These symbols are normally used to name memory locations, which may then be referenced by name. An assembly listing may be produced.

Digital-8-4-S

DDT-8

Dynamic Debugging Tape provides a means for on-line program debugging at the symbolic or mnemonic level. By typing commands on the console teleprinter, memory locations can be examined and changed, program tapes can be inserted, selected portions of the program can be run, and the updated program can be punched.

Digital-8-5-S

Floating-Point System

A Basic System
B Interpreter, I/O, I/O Controller
C Interpreter, I/O, Functions
D Interpreter, I/O, I/O Controller, Functions

Includes Floating-Point Interpreter and I/O subsystems. Allows the programmer to code his problem in floating-point machine language.

Floating-point operations automatically align the binary points of operands, retaining the maximum precision available by discarding leading zeros. In addition to increasing accuracy, floating-point operations relieve the programmer of the scaling problems common in fixed-point operations. This system includes elementary function subroutines programmed in floating-point. These subroutines are sine, cosine, square root, logarithm, arctan, and exponential functions. Data being processed in floating-point is maintained in three words of memory (12-bit exponent, 24-bit mantissa). An accuracy of seven decimal places is maintained.

Digital-8-6-S

Symbol Print

Loaded over the FORTRAN Compiler, this program lists the variables used and where they will be located in core. It also indicates the section of core not used by the compiled program and data.

Digital-8-7-S

DECtape Library System

The PDP-8 DECtape Library System is loaded by a $17_{10}$ instruction bootstrap routine that starts at $7600_8$. This loader calls a larger program into the last memory page, whose function is to preserve on tape the contents of memory from $6000_8$-$7577_8$, and then to load the INDEX program and the directory into those same locations. Since the information in this area of memory has been preserved, it can be restored when operations have been completed. The skeleton system tape contains the following programs:

| | |
|---|---|
| INDEX | Typing this causes the names of all programs currently on file to be typed out. |

| | |
|---|---|
| UPDATE | Allows the user to add a new program to the files. UPDATE queries the operator about the program's name, its starting address, and its location in core memory. |
| GETSYS | Generates a skeleton library tape on a specified DECtape unit. |
| DELETE | Causes a named file to be deleted from the tape. |

Starting with the skeleton library tape, the user can build up a complete file of his active programs and continuously update it.

Digital-8-8-S

MACRO-8

The MACRO-8 Symbolic Assembler accepts source programs written in symbolic language and translates them into binary form in two passes. MACRO-8 produces an object program tape (binary), a symbol table (for use with DDT), and octal symbolic assembly listing, and useful diagnostic messages. MACRO-8 is compatible with PAL III, and has the following additional features: user-defined macros; double precision integers, floating-point constants, arithmetic and Boolean operators, literals, text facilities, and automatic Link generation.

Digital-8-10-S

CALCULATOR

CALCULATOR is an equation evaluation routine. It differs from FORTRAN in that the function to be evaluated is entered via keyboard and calculated immediately upon termination of entry. Format control is provided so that computed results may be conveniently tabulated. Expressions causing the calling of common function subroutines are included.

Digital-8-11-S

DATAK

The DATAK system permits a complex, program-controlled data acquisition system to be adapted to a particular experimental environment through the use of a sophisticated and concise pseudo code. In addition to data-acquisition applications, DATAK furnishes the experimenter with a means of calibrating transducers and is a powerful aid in troubleshooting a complex data-gathering system. Paper tape output produced is acceptable as FORTRAN input.

Digital-8-12-S

ODT-II

ODT-II (Octal Debugging Tape) aids in debugging a PDP-8 program by facilitating communication with the program being run via the ASR 33 Teletypewriter. ODT-II features include register examinations and modification, control transfer, word searching, octal dumping, and instruction traps.

Digital-8-13-S

One-Dimensional Display and Analysis

The one-dimensional pulse-height analysis program is used to read in and analyze 1024-channel energy spectra data. The program receives and executes commands from the keyboard. These commands start and stop data taking and determine into which data region it goes, displays the data with markers, allows areas of interest on the display screen to be expanded, integrates between markers, writes out data, punches out data, and controls background subtraction.

Digital-8-14-S

Multiparameter Display and Analysis

The two-dimensional pulse-height analysis program is used to read in and analyze two-parameter energy and spectra data. The program receives and executes commands from the keyboard. These commands start and stop data taking, control the displays, and control writing and punching of the data. The displays available are: isometric, vertical and horizontal slicing, differential and integral contours, and "twinkle box." The program is flexible with respect to the dimensions of the data matrix.

Digital-8-15-S

Oceanographic Analysis

This program represents the basic accepted physical oceanography method for the reduction of data concerning depth, temperature, and salinity measurements of the water column.

This program has been designed to allow the field oceanographer a rapid means of immediately calculating Sigma-T, anomaly of specific volume, and sound velocity following a Nansen cast whereby he may examine in detail the results of his endeavor, to determine not only the structure of the environment he has just sampled but also to check the validity of his measurements.

In addition to the above, an interpolation routine is incorporated into the program as well as a depth integration of the anomaly of specific volume.

Digital-8-16-S

Master Tape Duplicator

The tape duplicator for the PDP-8 is a single-buffered read and punch program, utilizing the program interrupt. It computes a character count and checksum for each tape and compares with checks at the end of the tape.

Digital-8-35-S

A 680 5-Bit Character Assembly Subroutines
B 680 8-Bit Character Assembly Subroutines

These subroutines concentrate Teletype data by assembling serial-bit data into 5-bit (8-35-S-A) or 8-bit (8-35-S-B) characters and presenting the user with line number and character data. They also add start and stop bits and transmit characters serially. Full-duplex lines are assumed, but the subroutines will work with half-duplex if the user handles the expected echo.

## Elementary Function Routines

Digital-8-9-F

Square Root Subroutine - Single Precision

Forms the square root of a single-precision number. An attempt to take the square root of a negative number will give 0 for a result.

Digital-8-11-F

Signed Multiply Subroutine - Single Precision

Forms a 22-bit signed product from 11-bit signed multiplier and multiplicand.

Digital-8-12-F

Signed Divide Subroutine - Single Precision

This routine divides a signed 11-bit divisor into a signed 23-bit dividend giving a signed 11-bit quotient and a remainder of 11 bits with the sign of the dividend.

Digital-8-13-F

Double-Precision Multiply Subroutine - Signed

This subroutine multiplies a 23-bit signed multiplicand by a 23-bit signed multiplier and returns with a 46-bit signed product.

Digital-8-14-F

Double-Precision Divide Subroutine - Signed

This routine divides a 23-bit signed divisor into a 47-bit signed dividend and returns with a 23-bit signed quotient and a remainder of 23 bits with the sign of the dividend.

Digital-8-16-F

Sine Routine - Double Precision

The Double-Precision Sine Subroutine evaluates the function Sin(X) for $-4 < X < 4$ (X is in radians). The argument is a double-precision word, 2 bits representing the integer part and 21 bits representing the fractional part. The result is a 23-bit signed fraction $-1 < \text{Sin}(X) < 1$.

Digital-8-18-F

Cosine Routine - Double Precision

This subroutine forms the cosine of a double-precision argument (in radians). The input range is $-4 < X < 4$.

Digital-8-20-F

Four-Word Floating-Point Package

This is a basic floating-point package that carries data as three words of mantissa and one word of exponent. Common arithmetic operations are included as well as basic input/output control. No functions are included.

Digital-8-21-F

Signed Multiply (Uses EAE Type 182) Single Precision

This subroutine forms a 22-bit signed product from an 11-bit signed multiplier and multiplicand using the Extended Arithmetic Element Type 182. It occupies less storage and takes less time to execute than its non-EAE counterpart (Digital-8-11-F-Sym), and it has the same calling sequence.

Digital-8-22-F

Signed Divide (Uses EAE Type 182) Single Precision

This subroutine divides a double-precision signed 22-bit dividend by a signed 11-bit divisor, producing a signed 11-bit quotient and a remainder of 11 bits having the sign of the dividend.

It makes use of the Extended Arithmetic Element Type 182 instruction set and occupies less storage and takes less time to execute than its non-EAE counterpart Digital-8-12-F. It has the same calling sequence except that the subroutine name is changed from DIVIDE to SPDIV.

Digital-8-23-F

Signed Multiply (Uses EAE Type 182) Double Precision

This subroutine multiplies a 23-bit, signed 2's complement binary number by a 23-bit signed 2's complement binary number, giving a 46-bit product with two signs on the higher order end. It makes use of the Extended Arithmetic Element Type 182 instruction set and, because of this, occupies less storage and takes less time to execute than its non-EAE counterpart (Digital-8-13-F). Its calling sequence is compatible with the non-EAE version.

Digital-8-25-F

EAE Floating-Point Package

These packages perform the same tasks as the Floating-Point Packages (Digital-8-5-S A, B, C, D) except that certain routines have been speeded up by the use of the Extended Arithmetic Element Type 182.

For a detailed description of PDP-8 floating-point arithmetic and the Interpretive Floating-Point Packages, the reader is referred to Digital-8-5-S.

## Utility Programs

Digital-8-0

Format for PDP-8 Program Documentation

With the advent of the PDP-8, Digital Equipment Corporation introduced a new, recursive format for program documentation. This format is used for routines and subroutines, such as utility and functional, but not necessarily for system programs.

This format and its use are described in this document.

Digital-8-1-U

Read-In-Mode Loader

The RIM Loader is a minimum-sized routine for reading and storing the information in Read-In-Mode coded tapes via the ASR 33 Perforated Tape Reader.

Digital-8-2-U

Binary Loader (33ASR, 750, 183 Memory Extension)

The Binary Loader is a short routine for reading and storing the information in binary-coded tapes via the ASR 33 Perforated Tape Reader or by means of the Type 750 High-Speed Perforated Tape Reader.

The Binary Loader will accept tapes prepared by the use of PAL (Program Assembly Language; see Digital-8-3-S) or MACRO-8 (see Digital-8-8-S). Diagnostic messages may be included on tapes produced when using either PAL or MACRO. The Binary Loader will ignore all diagnostic messages.

Digital-8-3-U

DECtape Library System Loader

The use of the DECtape Library System Loader is discussed. Certain conventions with respect to last page storage are established for this loader as well as for the Read-In-Mode and Binary Loaders.

Digital-8-4-U-RIM

Read-In-Mode Punch ASR 33

This program provides a means of punching out the information in selected blocks of core memory as RIM-coded tape via the ASR 33 Perforated Tape Reader.

Digital-8-5-U

Binary Punch 33/75A

This program provides a means of punching out the information in selected blocks of core memory as binary-coded tape via the ASR 33 Perforated Tape Punch or via the High-Speed Punch 75A.

Digital-8-6-U

Octal Memory Dump

This routine reads the console switches to obtain the upper and lower limits of an area of memory, then types on the Teletype an absolute address plus the octal contents of the first four words specified and repeats this until the block is exhausted, at which time the user may repeat the operation.

Digital-8-7-U

Logical Subroutines

Subroutines for performing the logical operations of inclusive and exclusive OR are presented as a package.

Digital-8-8-U

Shift Right, Shift Left Subroutines (Single and Double Precision)

Four basic subroutines, shift right and shift left, each at both single and double precision, are presented as a package.

Digital-8-9-U

Logical Shift Subroutines

Two basic subroutines, shift right at both single and double precision, are presented as a package. The shifts are logical in nature.

Digital-8-10-U

Binary-Coded-Decimal to Binary Conversion Subroutine

This basic subroutine converts unsigned binary-coded-decimal numbers to their equivalent binary values.

Digital-8-11-U

Double Precision BCD-to-Binary by Radix Deflation

This subroutine converts a 6-digit BCD number to its equivalent binary value contained in two computer words.

Digital-8-12-U

Incremental Plotter Subroutine

This subroutine moves the pen of an incremental plotter to a new position along the best straight line. The pen may be raised or lowered during the motion.

Digital-8-14-U

Binary to Binary-Coded-Decimal Conversion

This subroutine provides the basic means of converting binary data to binary-coded-decimal (BCD) data for typeout, magnetic tape recording, etc.

Digital-8-15-U

Binary-to-Binary-Coded-Decimal Conversion (Four Digit)

This subroutine extends the method used in Digital-8-14-U so that binary integers from 0 to 4095 in a single computer word may be converted to four binary-coded-decimal characters packed in two computer words.

Digital-8-17-U

EAE (Type 182) Instruction Set Simulator

This routine permits the automatic multiply-divide hardware option to be simulated on a basic PDP-8.

Digital-8-18-U

Subroutine for Alphanumeric Message Typeout

This is a basic subroutine to type messages packed in computer words. Two 6-bit characters are packed internally in a single word. All ASR 33 codes from 301 to 337 and from 240 to 277 (excepting 243 and 245) can be typed. The typing of line feed (code 212) and carriage return (code 215) are made possible by arbitrarily assigning internal codes of 43 and 45, respectively, to represent these characters, thus preventing the output of ASCII codes 243 (#) and 245 (%).

Digital-8-19-U

Teletype Output Subroutines

A group of subroutines useful in controlling ASR 33 output is presented as a package. Provision is made for simulation of tabulation stops. The distance "tabbed" may be controlled by the user. Characrers whose ASR 33 codes are in the groups 241 through 277, inclusive, and 300 through 337, inclusive, are legal. Space, carriage return then line feed, and tabulation are provided via subroutines.

Digital-8-20-U

Character String Typeout Subroutine

This basic subroutine types messages stored internally as a "string" of coded characters. All ASR 33 characters are legal.

Digital-8-21-U

Symbolic Tape Format Generator

The Format generator allows the user to create PDP-8 symbolic tapes with Formatting. It may be used to condense tapes with spaces by inserting tabs, or merely to align tabs, instructions, and comments.

Digital-8-22-U

Unsigned Decimal Print

This subroutine permits the typeout of the contents of a computer word as a 4-digit, positive, decimal integer.

Digital-8-27-U

DECtape Subroutines

Allows the programmer to read, write, or search DECtape using prewritten and tested subroutines. A series of subroutines which will read or write any number of DECtape blocks, read any number of 129-word blocks as 128 words (or one memory page), or search for any block (used by read and write, or to position the tape). These programs are assembled with the user program and are called by a jump to subroutine instruction. The program interrupt detects the setting of the DECtape (DT) flag, allowing the main program to proceed while the DECtape operation is being completed. A program flag is set when the operation is completed. The program thus effectively allows concurrent operation of several input/output devices with the DECtape.

Digital-8-32-U

Binary Punch (6 Channel)

This program provides a means of punching out the information in selected blocks of core memory as binary-coded tape via the 6-channel high-speed punch.

Digital-8-33-U

5/8 TOG (DECtape Formatter)

This program is designed to write timing tracks, mark tracks, and block numbers onto a reel of DECtape providing the tape with the basic skeletal format necessary for its inclusion in any programmed DECtape system. The Formatter program also performs preliminary read-data and write-data checks to assure the user that the tape produced can be reliably included in such an environment.

Digital-8-34-U

DECEX DECtape Exerciser

This program provides complete certification of the DECtape format produced.

## Maintenance Programs

Maindec 801-1

PDP-8 Instruction Test Part 1

This program is a minimal test of memory reference instructions, operate instructions, interrupt mode, and the keyboard printer. This test should be used when the state of the processor prevents read-in

of more advanced diagnostic programs. It is simply a "go-no go" test of the instructions and is not intended to be diagnostic.

Maindec 801-2A

PDP-8 Instruction Test Part 2A

This program is a test of memory reference instructions, operate instructions, and interrupt mode. An attempt is made to detect and isolate errors to their most basic faults and to the minimum number of logic cards.

Maindec 801-2B

PDP-8 Instruction Test Part 2B

This program is a test of TWOS ADD (TAD) and ROTATE logic, (RAL, RTL, RAR, RTR). Random numbers are used in the TWOS ADD portion of the test and sequential numbers are used in the ROTATE portion. Program control is dependent upon operator manipulation of four switches in the SWITCH REGISTER (bits 0, 1, 2, 3). Error information is normally printed out on the keyboard printer.

Maindec 801-2C

PDP-8 JMS and JMP Test

This program tests the JMP and JMS instructions by doing a JMP and JMS to locations 177-4000. The program also tests the JMS return address for accuracy.

Maindec 801-3A

PDP-8 Instruction Test (EAE Type 182) Part 3A

This program is a test of the Extended Arithmetic Element Type 182. The following instructions are tested: MQL, MQA, SHL, LSR, ASR, NMI, SCA. An attempt is made to detect and isolate errors to their most basic faults and to the minimum number of logic cards. Multiply and divide are tested by Maindec 801-3B.

Maindec 801-3B

PDP-8 Instruction Test (EAE Type 182) Part 3B

Divide overflow detection hardware and divide and multiply hardware are tested by using a pseudo random-number generator to produce the parameters for each test. A software simultated divide and multiply are used to test the results of the hardware divide and multiply.

Maindec 802

Memory Checkerboard Test

Maindec 802 tests memory for core failure on half-selected lines under the worst possible conditions for reading and writing. It is used primarily for testing the operation of memory at marginal voltages.

There are two versions of Maindec 802. The Low End program occupies registers 0003-0111 octal and tests memory from 0112-7777 octal. The High End program occupies registers 7450-7555 octal and tests memory from 0000-7447 octal.

Maindec 803

PDP-8 Memory Address Test

Maindec 803 is designed to provide rough inspection of the performance of the Memory Ad-

dress register and the decoder network which selects a given memory cell. It is used primarily to detect errors arising from open or shorted selection lines.

Maindec 810

PDP-8 Teletype Reader Test

Maindec 810 tests performance of the Teletype Model 33 Perforated Tape Reader using the reader to scan a closed-loop test tape punched with alternating groups of character codes 000 and 377.

Each character is tested for bits dropped or gained while reading; each group of characters is checked for characters missed entirely or read more than once.

Maindec 811

PDP-8 High Speed Reader Test

This program tests performance of the Type 750 High Speed Perforated Tape Reader and control by scanning a closed-loop test tape for transmission accuracy. The reader control is tested for correct operation with the PDP-8 interrupt system.

Maindec 812

PDP-8 Teletype Punch Test

Maindec 812 punches a test tape in a predetermined pattern. The tape passes directly from the Teletype punch to the Teletype reader, which checks the pattern for accuracy.

Maindec 814

PDP-8 Teleprinter Test

The PDP-8 Teleprinter Test tests performance of the Teletype 33 Keyboard Printer. There are two parts to the test, selectable by the operator. The first part tests keyboard input by immediately causing the character typed to be printed for comparison. The second part tests continuous operation of the teleprinter by causing a line consisting of the ASCII character set to be repeatedly printed. The latter also tests for correct functioning of the interrupt after a character has been printed.

Maindec 817

PDP-8 High Speed Punch Test

This program consists of two separate tests. The first causes the High Speed Punch Type 75E to produce a tape containing a sequence of "pseudo-random" character codes. This tape is checked for accuracy using either the high-speed reader or the Teletype reader.

In the second test, the character code represented by the setting of $SR_{4-11}$ is punched repeatedly. The switch setting may be changed while the test is running.

Maindec 820-1

Extended Memory Control Part 1

This program exercises and tests Extended Memory Type 183 instructions CDF, CIF, RDF, RIF, RMF, and RIB, for proper operation. Basically, this program tests the control section of the Type 183 memory. Data is tested by tests Maindec 802 and Maindec 820-2.

Maindec 820-2

Extended Memory Checkerboard Part 2

Maindec 820-2 is a preliminary test for core memory failures on half-selected lines under

worst-case conditions of reading and writing. It is used to test memory module X while running the program in memory module Y.

Maindec 825

680 Static Test

The 680 Static Test verifies correct operation of the 681 and 685 circuits associated with the 680 Data Communications System, in a static state. That is, the program does not actually transmit characters, but tests only the logical operation of the hardware. Hardware malfunctions detected by the program result in a processor halt.

Maindec 826

A 680 8-Bit Character Exerciser

B 680 5-Bit Character Exerciser

The 680 Character Exerciser Program further verifies correct operation of the 680 Data Communications System. This test assumes that the Teletype lines are full duplex. However, if the line outputs are jumpered to the line inputs, the test does verify that the input characters are received as transmitted.

Maindec 827

580 Utility Routines and Compiler

This program is designed to exercise the 580 Tape System. The test routines are called from a small compiler, and are under control of a pseudo language, which may be stored on paper tape for daily maintenance, or typed on-line for debugging and observing malfunctions of the 580 Tape System.

Maindec 827-U

Magnetic Tape Type 580 Utility Routines

These subroutines allow the user to operate the 580 Magnetic Tape System by providing most commands associated with a more sophisticated (hardware) tape system.

Maindec 828

PDP-8 LT08 Teleprinter Test

The LT08 Teleprinter Test verifies correct operation of the LT08 Control Line hardware and any configuration of from one to five teleprinters. Hardware malfunctions detected by the program result in a processor halt. The test includes a Concurrent Output Routine, a Concurrent Input Routine, an Output Scope Loop, and a WRU Test that verifies that none of the teleprinters associated with the LT08 respond to a WRU (who are you) code.

Maindec 829

PDP-8 Memory Power On/Off Test

This program tests memory for bit drop out and pick up after a simulated power failure.

Maindec 830

Type 30G Symbol Generator Exerciser

This program exercises symbol generator logic by using selected character patterns.

Maindec 831

PDP-5/8 DECtape Maintenance Package

The PDP-5/8 DECtape Maintenance Package is a collection of routines designed to be used by maintenance personnel as aids in debugging hardware troubles and as periodic confidence checks on correct operation of the device. Routines are provided to test IOT instructions, delays, control registers, timing, and basic modes of operation. Other routines are included which allow the operator to adjust the device more efficiently and exercise different modes of operation pursuant to "scoping" machine functions. Routines to obtain octal dumps of core memory and routines to write varying bit patterns in core are also available.

Maindec 832

Real-Time Clock Test

This program tests the real-time clock IOT logic, and crystal oscillator specifications.

Maindec 833

Lots of Little Pictures on the Eight

This program contains 12 individual 338 buffered display routines. The routines were selected to enable adjustment and validation of CRT Analog/Digital hardware.

Maindec 834

Type 338 Display PJMP Test

This program is a test of the PJMP instruction. On the 338 display analyses of the Pushdown Pointer, Display Address Counter, Status, Push Jump Destination and Return Addresses are printed upon detection of an error in these areas.

Maindec 835

Type 338 Display POP Test

This program is a test of the POP instruction. On the 338 Display analyses of the Display Address Counter and Pushdown Pointer are printed upon error detection.

Maindec 839

PDP-8 Memory Parity Option

This program is designed to exercise and detect memory parity control and data errors on the PDP-8.

DECUS No. 5-1 and 5-2

Service and Debugging Subroutines for the PDP-5

Two basic subroutine packages for use in communicating with the PDP-5 for service and debugging functions have been written at Bell Telephone Laboratories. The first package, called the Binary Package, is a completely independent one-page subroutine for handling paper tape input-output. The second, called the Octal Package, is a two-page set of subroutines which facilitates exchange of information between the operator and the computer via the Teletype 33ASR unit. As an addition to the Octal Package, there is a symbolic instruction dump subroutine package which occupies three pages of storage.

1. The Binary Package contains a BIN format loader, and provisions for dumping of bracketed locations in BIN or RIM format, punching leader trailer, punching a check sum on BIN tapes, and punching a starting address for self-starting RIM tapes. If entered as a subroutine, the appropriate return can be made. Control of the package is made by the switch register and CONTINUE switch on the PDP-5 control panel.

2. The Octal Package contains provisions for an eight to the line octal dump of bracketed locations, moving of blocks of information in the memory, and loading of memory from the Teletype. Links to the Binary Package and the symbolic dump are also included. Control for this package is through the Teletype and the switch register.

   The symbolic dump portion of the Octal Package provides for a printing on the Teletype of up to four pieces of information for each location of a bracketed group of locations. These include the address, the octal contents of that address, the interpretation as two trimmed Teletype code characters of the octal contents and the symbolic instruction decoding of the contents. Any combination of these pieces of information can be selected through use of the switch register.

DECUS No. 5-3

BRL - A Binary Relocatable Loader with Transfer Vector Options for the PDP-5 Computer

BRL is a binary loader program occupying $4640_8$ to $6177_8$ registers: also 160 to 177. It has two main functions:

1. It allows a PDP-5 operator to read a suitably prepared binary program into any page location in memory except the registers occupied by BRL. Thus, a program need not be reassembled from symbolic to binary tape whenever it is desired to relocate it.

2. It greatly simplifies the calling of programmed subroutines by allowing the programmer to use an arbitrary subroutine calling sequence when writing his program, instead of having to remember the location of the subroutines. For example, if a programmer wishes to call "Square Root," he does not have to know where Square Root is stored in memory; BRL will instead keep track of this for him. This feature is available to the IBM 7094 programmer and is known as a "transfer vector" option.

DECUS No. 5-4

Octal Typeout of Memory Area with Format Option

Write-up and Listing Only

DECUS No. 5-5

Expanded Adding Machine

Expanded Adding Machine is a minimum-space version of Expensive Adding Machine (DEC-5-43-D) using a table lookup method and including an error space facility.

This is a basic version to which additional control functions can easily be added: Optional vertical or horizontal format, optional storage of intermediate result without reentry, fixed-point output of results within reason, and other features that can be had in little additional space under switch register control.

Write-up and Listing Only

DECUS No. 5-6

BDC to Binary Conversion of 3-Digit Numbers

This program is based on DEC-5-4 and is intended to illustrate the use of alternative models in program construction.

While not the fastest possible, this program has one or two interesting features. It converts any 3-digit BCD-coded decimal number, $D_1D_2D_3$ into binary in the invariant time of 372 microseconds. Efficient use is made of BCD positional logic to work the conversion formula $(10D_1 + D_2)\ 10 + D_3$ by right shifts in the accumulator. In special situations, it could be profitable to insert and initial test/exit on zero, adding 12 microseconds to the time for non-zero numbers.

Write-up and Listing Only

DECUS No. 5/8-7

Decimal to Binary Conversion by Radix Deflation on PDP-8

Write-up and Listing Only

DECUS No. 5-8

PDP-5 Floating Point Routines

Consists of:

1. Square Root - Tape and Symbolic Listing
2. Sine-Cosine - Tape
3. Exponential - Tape

DECUS No. 5/8-9

Analysis of Variance PDP-5/8

The analysis of variance program was written for the standard PDP-5/8 configuration (i.e. 4K memory, ASR33 teletype). The output consists of:

A. For each sample:
   1. Sample number
   2. Sample size
   3. Sample mean
   4. Sample variance
   5. Sample standard deviation

B. The grand mean

C. Analysis of Variance Table:
   1. The grand mean.
   2. The weighted sum of squares of class means about the grand mean.
   3. The degrees of freedom between samples.
   4. The variance between samples.
   5. The pooled sum of squares of individual value about the means of their respective classes.

6. The degrees of freedom within samples.
7. The variance within samples.
8. The total sum of squares of deviations from the grand mean.
9. The degrees of freedom.
10. The total variance.
11. The ratio of the variance between samples to the variance with samples.

This is the standard analysis of variance table that can be used with the F test to determine the significance, if any, of the differences between sample means. The output is also useful as a first description of the data.

All arithmetic calculations are carried out by the Floating Point Interpretive Package (Digital-8-5-S).

DECUS No. 5-10

Paper Tape Reader Test

A test tape can be produced and will be continuously read as an endless tape. Five kinds of errors will be detected and printed out. The Read routine is in 6033-6040. Specifications: Binary with Parity Format - Length: registers in locations (octal): 10, 11, 40 through 67 (save 63, 64), and 6000-7777.

DECUS No. 5-11

PDP-5 Debug System

Purpose of this program is to provide a system capable of:

1. Octal dump 1 word per line.
2. Octal dump $10_8$ words per line.
3. Modifying memory using the typewriter keyboard.
4. Clearing to zero parts of memory.
5. Setting to HALT codes part of memory.
6. Entering breakpoints into a program.
7. Initiating jumps to any part of memory.
8. Punching leader on tape.
9. Punching memory on tape in RIM format.
10. Punching memory on tape in PARITY format.
11. Load memory from tape in PARITY format.

DECUS No. 5-12

Pack-Punch Processor and Reader for the PDP-5

The processor converts a standard binary-format tape into a more compressed format, with two twelve-bit words contained on every three lines of tape. Checksums are punched at frequent intervals, with each origin setting or at least every 200 words.

The reader, which occupies locations 7421 to 7577 in the memory, will load a program which is punched in the compressed format. A test for checksum error is made for each group of 200 or less words, and the program will halt on detection of an error. Only the most recent group of words will have to be reloaded. Read-in time is about ten percent less than for conventional binary format, but the principal advantage is that little time is lost when a checksum error is detected, no matter how long the tape.

DECUS No. 5-13

PDP-5 Assembler

This program accepts symbolic programs punched on cards and assembles them for the PDP-5. An assembly listing is produced, and a magnetic tape is generated containing the program. This magnetic tape can be converted to paper tape and then read into the PDP-5, or it can be read directly into a PDP-5 with an IBM compatible tape unit. Cards are available.

DECUS No. 5/8-14

Dice Game for the PDP-5, PAL

Binary tape and write-up only. Program uses program interrupt facility.

DECUS No. 5-15

ATEPO (Auto Test in Elementary Programming and Operation of a PDP-5 Computer)

The program will type questions or instructions to be performed by the operator of the PDP-5 (4K) computer. The program will check to see if the operator has followed the instructions or has answered the questions correctly. If this is the case, it will type the next question or instruction.

The program itself uses the locations $200_8$ to $500_8$, and the messages to be typed are in $600_8$ -$3410_8$. The area $500_8$ -$577_8$ is used to store the RIM and BIN loaders while the programming is running. Page 0 and the locations above $4000_8$ are a "work area" in which all the instructions are going to be executed, and in which the operator can make practically all kinds of mistakes, because the contents of the locations in that area are reset after typing any message. The program requires the RIM and BIN loaders to be in locations $7700_8$-$7777_8$ in order to transfer to the "safe area."

After using the program, the loaders can be returned to their original position by just starting the computer in location 377; it will jump to a small subroutine in $3500_8$-$3515_8$ that will make the transfer. The possibility of mistakes that would interfere with the program itself is reduced to practically zero, if the bit 0 is permanently kept (except when answering question 4) in the position 1. With the exception mentioned above, all of the other solutions can be accomplished with the bit 0 in position 1. For that reason, we strongly recommend that a piece of tape, or something similar, be put over the switch corresponding to that bit when being used by an operator without experience, for whom the program was designed.

DECUS No. 5/8-16

Tape Duplicator for the PDP-5/8

The tape duplicator for the PDP-5/8 is a single buffered read and punch program utilizing the program interrupt. It computes a character count and checksum for each tape and compares with checks at the end of the tape.

Checks are also computed and compared during punching. There are three models of operation:

A. SWITCH Ø ON - MAKE MASTER TAPE
B. SWITCH 1 ON - DUPLICATE MASTER TAPE
C. SWITCH 2 ON - VERIFY DUPLICATION

During duplication, the program will notify the operator whether or not more copies can be made without re-reading the master.

Binary and symbolic tapes are available.

DECUS No. 5/8-17

Type 250 Drum Transfer Routine For Use on PDP-5/8

Transfer data from drum to core (Read) or core to drum (Write) via ASR-33 Keyboard Control.

DECUS No. 5-18

Bin Tape Disassembler for the PDP-5*

This program disassembles a PDP-5 program, in Bin format, on punched paper tape. The

*Work performed under the auspices of the U. S. Atomic Energy Commission

tape is read by a high-speed reader, but the program may be modified to use the ASR-33 reader. The margin setting, address, octal contents, mnemonic interpretation (PAL), and the effective address are printed on the ASR-33 Teletype.

DECUS No. 5-19

DDT-5-2 Octal-Symbolic Debugging Program

DDT-5-2 is an octal-symbolic debugging program for the PDP-5 which occupies locations 5600 through 7677. It is able to merge a symbol table punched by PAL II and stores symbols, 4 locations per symbol, from 5577 down towards 0000. The mnemonics for the eight basic instructions and various OPR and IOT group instructions are initially defined (see DEC-5-1-S Attachment II, p. 21), and the highest available location for the user is initially 5373.

From the teletype, the user can symbolically examine and modify the contents of any memory location. DDT-5 allows the user to punch a corrected program in BIN format.

DDT-5 has a breakpoint facility to help the user run sections of his program. When this facility is used, the debugger also uses location 0005.

This program has nearly all the features of DDT for the PDP-1. The meaning of the control characters of ODT (DEC-5-5-S) are the same in DDT-5.

DECUS No. 5/8-20

Remote Operator FORTRAN System

Program modifications and instructions to make the FORTRAN OTS version dated 2/12/65 operated from remote stations.

DECUS No. 5/8-21

Triple Precision Arithmetic Package for the PDP-5 and the PDP-8

This is an arithmetic package to operate on 36-bit signed integers. The operations are add, subtract, multiply, divide, input conversion, and output conversion. Triple precision routines have a higher level of accuracy for work such as accounting. The largest integer which may be represented is $2^{35}-1$ or 10 decimal digits. The routines simulate a 36-bit (3 word) accumulator in core locations 40, 41, and 42 and a 36-bit multiplier quotient register in core locations 43, 44, and 45.

Aside from the few locations in page 0, the routines use less core storage space than the equivalent double-precision routines.

DECUS No. 5/8-22

DECtape Duplicate

This is a DECtape routine to transfer all of one reel (transport 1) to another (transport 2). This program occupies one page of memory beginning at 7400. The last page of memory is not used during the operation of the program, however, the memory from 1 to 7436 is used to set the DECtape reels in the proper starting attitude and is then destroyed during duplication. Duplication will commence after which both reels will rewind. Parity error will cause the program to halt with 0040 in the accumulator.

DECUS No. 5/8-23

PDP-5/8 Oscilloscope Symbol Generator

The subroutine may be called to write a string of characters, a pair of characters, or a single character on an oscilloscope. Seventy (octal) symbols in ASCII Trimmed Code and four special "format" commands are acceptable to this routine. The program is operated in a fashion similar to the DEC Teletype Output Package.

Binary tape with parity format, PAL binary tape, Assembler listing, and cards for an LRL Assembly are available.

Specifications:
1. BIN with parity format or PAL BIN
2. Length - registers 200-577 (octal)
3. Oscilloscope display unit

DECUS No. 5-24

Vector Input/Edit

This program accepts Teletype and effects editing options by implementing a man-machine dialogue. Development of the program was supported, in part, by the Air Force Office of Scientific Research and the Army Research Office.

DECUS No. 5-25

A Pseudo Random Number Generator for the PDP-5 Computer

The random number generator subroutine, when called repeatedly, will return a sequence of 12-bit numbers which, though deterministic, appears to be drawn from a random sequence uniform over the interval $0000_8$ to $7777_8$. Successive numbers will be found to be statistically uncorrelated. The sequence will not repeat itself until it has been called over 4 billion times.

The program tape is prefixed with a text for a relocatable loader, used at NYU, but this may be bypassed and the binary section will then read directly into 3000-3077.

DECUS No. 5-26

Compressed Binary Loader (CBL) Package

PDP-5 Installations using an ASR-33 Teletype for reading in binary tapes can save significant time (approximately 25%) by taking advantage of all eight channels of the tape. The CBL loader only occupies locations 7700 through 7777. The tape formatted into individual blocks, each with a checksum.

On detection of an error, the loader halts so the tape may be repositioned in the leader area of the block which caused the error.

PAL II has been modified to punch in CBL format, and a DDT-5-3 (comparable to DDT-5-5, DECUS No. 5-19) has been written.

The following programs are included in the package:

1. CBL Loader
2. CBC Converter (BIN to CBL)
3. CONV Converter (CBL to BIN)
4. PAL IIC (punches CBL format)
5. DDT-5-3 (reads and punches CBL format)

DECUS No. 5/8-27

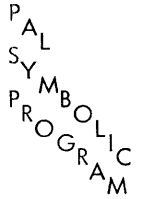
ERC Boot

The ERC Boot is a bootstrap routine somewhat simpler than the one presently available for the PDP-8. This routine restores the entire last page, consisting of:

1. Clear Memory Routine
2. RIM Loader
3. Modified Binary Loader.

The Clear Memory routine is entered at 7600 (octal). It clears (to 0000) the lower 31 pages of memory, then branches to the Binary Loader.

The modified Binary Loader halts after reading tape with the checksum in the accumulator. If the binary tape is properly terminated, pressing CONTINUE takes a branch to the beginning location of the program. PAL compiled programs may be properly terminated by ending the PAL symbolic tape in the following manner:




*START (any named starting address)
$

The Binary Loader stores the octal value for START in the location labeled ORIGIN in BIN. The instruction following HLT in BIN is replaced by JMP I ORIGIN (5616), causing a branch to START.

DECUS No. 8-28

PAL III Modifications for PDP-8 and ASR-33

This modification of the PAL III Assembler speeds up assembly on the ASR-33/35 and operates only with this I/O device. The symbolic tape is read only once, on pass 0, and stored in the machine. This pass initiates as does pass 1, except that both switches 0 and 1 must be down. Other passes of the assembly initiate normally, but do not end the symbolic tape.

If the program is too large for the buffer, the Teletype punches 50-200 codes before halting. If the program size is doubtful, it is advisable to leave the punch on during pass 0 and not continue to pass 1.

The user and symbol table starts at 2736 and the buffer begins at 3103, allowing room for 25 user symbols only. The highest location of the buffer is 7440. This leaves a buffer size of $4336_8$ or $2270_{10}$, which is sufficient for a large program or a small program with many comments, owing one tape character per location. To increase symbol table size, constants at 1413 and 1414 may be adjusted. For instance, if 50 symbols are desired:

```
1413    BUF, 3246
1414    BUFCNT, -4172
```

A few other modifications have been made to aid in circumventing the slow speed of the Teletype. The length of the leader-trailer tape has been shortened; there is no pass III leader punched; and the symbol table has no leader or trailer on either pass, making the symbol tape incompatible with DDT (an appropriate leader can be punched manually). To restore any of these characteristics, the appropriate statements in the modifications tape may be deleted.

Symbolic tape may be modified as above and a new binary tape produced. The binary tape must then read in after loading the assembler for modification. This process can be done each time the assembler is loaded, or the modified assembler can be punched as a complete separate tape.

DECUS No. 8-29

BCD to Binary Conversion Subroutines

These two subroutines improve upon the DEC supplies conversion routine. Comparison cannot be made to the DECUS-supplied fixed-time conversion, DECUS No. 5-6, because it is specified only for the PDP-5. One routine is designed for minimal storage, the other for minimal time. Both are fixed-time conversions; time specified is for a 1.6-μsec machine.

Minimal time routine: 73.6 μsec/32 locations

Minimal storage routine: 85 μsec/29 locations

DEC routine: 64-237 μsec/37 locations

Time for number $D_1$, $D_2$, and $D_3$ is $64 + (D_1 + D_2)$ 9.6 μsec.

DECUS No. 5-30

GENPLOT - General Plotting Subroutine

This self-contained subroutine is for the PDP-5 with a 4K memory and a CALCOMP incremental plotter. The subroutine can move (with the pen in the up position) to location (x, y), make an "X" at this location, draw a line from this present position to location (x, y), and initialize the program location counters. A binary, relocatable tape is available.

If the subroutine is to be relocatable, the titles are: MOve, PLot, DRaw, and INit. The readin procedure is the same as for other relocatable subroutines.

DECUS No. 5-31

FORPLOT - FORTRAN Plotting Program for PDP-5

FORPLOT is a general-purpose plotting program for the PDP-5 computer in conjunction with the CALCOMP 560 Plotter. It is self-contained and occupies memory locations $0000_8$ to $4177_8$. FORPLOT accepts decimal data inputted on paper tape in either fixed or floating point formats. Formats can be mixed at will. PDP-5 FORTRAN output tapes are acceptable directly and any comments on these are filtered out.

FORPLOT scales input data. The operator informs the computer, in advance, of the data values to be assigned to the top, bottom, right, and left plot boundaries. There are no restrictions on these data values. It is not necessary that any of them be zero, nor is it necessary that the top and right boundaries correspond to more positive data values than the bottom and left boundaries, respectively.

All plotted graphs are 10 inches in the ordinate direction. The operator controls the length of the abscissa which may reach a maximum of 39.99 inches.

A number of plot format options are available to the user. The program is capable of drawing an abscissa and ordinate axis, each ticked at intervals of 1 inch, at the right and bottom boundaries of the plot, respectively. If the user chooses to omit these axes, a circled point is placed at the starting point to indicate the bottom-right plot boundary. Points are left unconnected unless connected through user control. Finally, at the option of the user, FORPLOT can locate either a small "x" or a small octagon (approximately 1/16 inch across), or a superposition of both at each plotted point to make the point more plainly visible.

A column selection feature is provided enabling the operator to select data columns from tapes containing several of them. In this bulletin "column" refers to a column in the arrangement of the data when it is printed on the ASR-33. If desired, the operator can supply only the ordinates and the program will space the plotted points uniformly in the abscissa direction according to a preset constant.

DECUS No. 5/8-32

Program to Relocate and Pack Programs in Binary Format

This relocation program allows automatic transfer of a program in binary format into any portion of memory, no matter what starting address it has been allocated. Each program is given a

fixed starting address, allowing it to be loaded in the normal manner without using the relocation program. This includes moving a program to an entirely different starting address on a different page of memory. The program is now in use at CRNL and is proving a most effective tool in assembling a combination of existing programs and in amending and fault-finding new programs.

This package includes three programs to aid in relocating and retrieving subroutines:

1. A Clear Memory routine which clears registers $200_8$ through $3777_8$. Length is 12 locations.

2. A programmed display routine which displays the entire memory. Empty registers appear on the base line while occupied addresses appears as 4096 counts on the display. The length is 15 locations.

3. Because it is often necessary to retrieve relocated programs on paper tape, a punch routine is included which punches those areas of the first half of memory which contain the program. The punch program senses and deletes gaps in memory.

   This program saves much time since starting and ending addresses need not be remembered as with ODT and other types of binary punch programs.

   The tape is punched in binary format complete with checksum, and is preceded by and followed with a length of leader. The length is 90 locations.

The relocation program and associated programs are themselves relocatable.

DECUS No. 5/8-33

Tape to Memory Comparator

Tape to Memory Comparator is debugging program which allows comparison of the computer memory with a binary tape. It is particularly useful for detecting reader problems, or during stages of debugging a new program.

A typeout occurs whenever the memory disagrees with the contents of the binary program tape. The typeout consists of the memory location, contents of memory, and contents of the tape on one line. The checksum is typed out as an error at one location greater than the last address on the tape.

Presently, the program uses a high-speed reader; however it may be modified for the TTY reader. The program occupies 165 octal locations on a single page. It does not use page 0 or autoindex registers.

DECUS No. 5-34

Memory Halt - A PDP-5 Program to Store Halt in Most of Memory

With Memory Halt and OPAK, (DECUS No. 5-2.1), in memory, it is possible to store halt (7402) in the following memory locations:

0000 to 0005
0007 to 6177
7402 and 7403

Memory Halt occupies locations 0200 to 0237 with a starting address of 0200. When started, it stores 7402 in locations 0001 to 0005 and locations 7402 and 7403. It then sets up some memory location so that OPAK can store 7402 in locations 0007 to 6177.

Halts in memory are useful when a program transfers control to an area of memory not occupied by the program itself. Upon executing the JMP or JMS instruction, the computer halts. With careful investigation, the programmer can determine why the transfer of control took place.

DECUS No. 5/8-35

Binary Coded Decimal to Binary Conversion Subroutine and Binary to Binary Coded Decimal Subroutine (Double Precision)

This program consists of a pair of relatively simple and straightforward double-precision conversions. They make no claim to speed or brevity. A double entry has been used which is:

```
TAD  HIGH
JMS  BCD BIN
TAD  LOW
JMS  BCD BIN+3
```

DECUS No. 5-36

Octal Memory Revised

The Octal Memory Dump on Teletype is a DEC routine (DEC-5-8-U) which dumps memory by reading the switch register twice; once for a lower limit and again for an upper limit. It then types an address, the contents of the program and the next three locations, issues a CR/LF, then repeats the process for the next four locations. This leaves the right two-thirds of the Teletype page unused. The $78_{10}$ instructions occupy two pages.

This revised routine uses the complete width of the Teletype page and occupies only one memory page, using less paper and two less instructions. Now an address and the contents of 15 locations are typed out before a carriage return.

Octal Memory Dump Revised has proved its value as a subroutine and/or a self-contained dump program when it is necessary to dump large sections of DECtape, magnetic tape (IBM compatible), or a binary formatted paper tape.

DECUS No. 5-37

Transfer II

For users who have more than one memory bank attached to the PDP-5/8, Transfer II may prove valuable in moving information from one field to another. Often areas designated for loaders are being used for other reasons, only to find the loaders necessary a few minutes later. When debugging, Transfer II enables a programmer to make a few changes in a new program and test it without reading in the original program again, especially if his corrections did not work. In short, Transfer II enables more extensive use of memory banks.