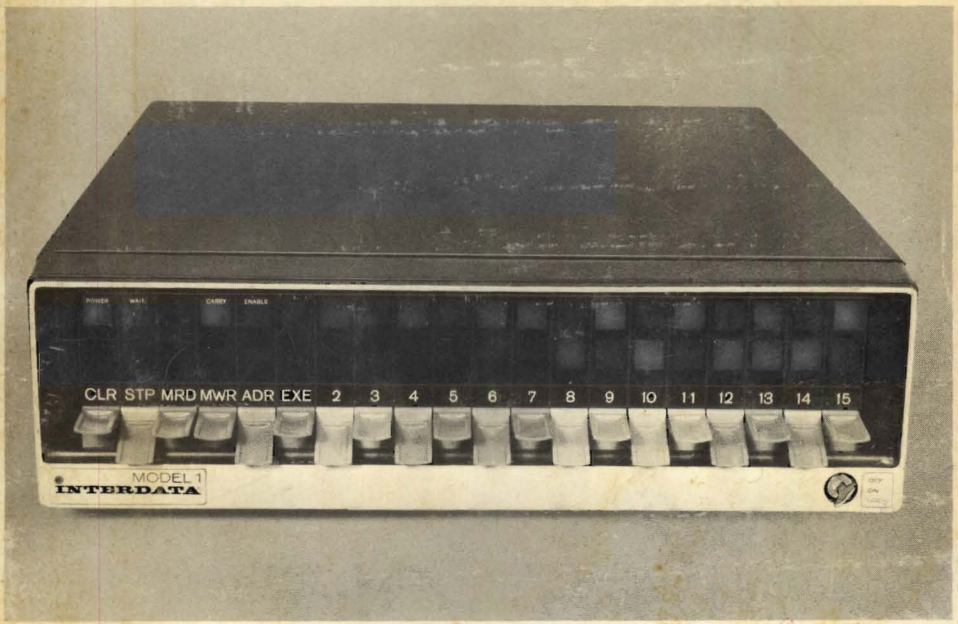
E.M. GETTEL

Model 1
User's Manual

INTERDATA®



E. M. GETTEL

Model 1
User's Manual

Publication Number 29-215

INFORMATION CONTAINED IN THIS MANUAL IS SUBJECT TO DESIGN CHANGE OR PRODUCT IMPROVEMENT

Printed in U.S.A. • September 1970

QUICK REFERENCE INDEX

To aid in quickly locating a particular chapter, the index marks on the edge of this page are aligned with similar marks on the first page of each chapter.

MODEL 1 USER'S MANUAL

MODEL 1 USER'S MANUAL (Cont)

Clay Archer

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

TABLES

CHAPTER 1
INTRODUCTION

1.1 GENERAL FEATURES

The INTERDATA Model 1 Processor is a physically small, high speed, application-oriented, modular processor designed to provide maximum computing power at minimum cost to the user. Modular construction is used for ease of maintenance and to facilitate system configuration expansion to meet the future needs of the user. The Model 1 Processor is excellent for applications in diverse fields such as industrial monitoring and control, process control, data collection and data communications.

The Model 1 Processor contains four hardware priority External Interrupt lines as standard items. Four additional External Interrupt Lines are available as an option. All interrupt lines are individually maskable. The Processor uses eight and sixteen-bit instructions for efficient coding and optimum core utilization. Many of the instructions contain test and skip options for effective loop control. Powerful bit manipulating instructions efficiently handle bit processing. A powerful auto-indexing feature enables the system to use any core location as an index register.

A Power Fail Safe option is provided to prevent data or program loss as a result of failures in the primary power source. Also at user option, the program can continue, halt, or execute a recovery routine when power has been restored.

1.2 MEMORY

The memory system uses highly reliable 2,048 byte core modules and is expandable to 16,384 bytes, with Parity as an option. The cycle time is one microsecond. Plug-compatible 2,048 byte, Read-Only-Memory modules may be intermixed with core modules for ultra-reliable, inexpensive, nonvolatile program storage. The memory system is organized into 256 byte pages. Two pages, the Current Page and Page Zero, are directly addressable by the primary instruction word. All remaining pages are addressed indirectly.

1.3 INPUT/OUTPUT

1.3.1 Multiplexor Bus

There are several ways for handling Input/Output transfers between the Model 1 Processor and up to 256 devices. The instruction set includes eight instructions for Input/Output in addition to Read Block, Write Block, and the Pulsed I/O instruction. The Read Block and Write Block instructions can transfer data at speeds up to 500,000

bytes per second. The Pulsed I/O instruction can specify any combination of three control pulses, thus providing a convenient and economical way for special interface design.

There is a serial I/O port on the standard Model 1 Processor which handles bit serial data streams such as those from Teletypewriters. The serial I/O port is interrupt driven and does not lock up the Processor when it is in use. The Processor uses the standard one millisecond real time clock to control the serial I/O port on Input or Output and to support other application oriented software.

The I/O Bus of the Model 1 Processor is hardware plug compatible with the I/O Bus of other INTERDATA Processors.

1.3.2 Direct Memory Access Channel (DMAC)

The Model 1 has a built-in Direct Memory Access port which accepts up to four DMA Channels. This port operates on a cycle-steal basis, allowing simultaneous I/O transfer. The maximum transfer rate is 1,000,000 bytes per second.

The customer can interface his own devices directly to this port or use one of the following INTERDATA devices:

1. Selector Channel
   Emulates the standard INTERDATA Multiplexor Bus for block transfers between memory and up to 25 standard INTERDATA devices.

2. Universal Direct Memory Access Interface
   Provides all of the front end logic to the memory bus, and provides wire-wrap IC locations (59) to simplify interfacing customer-designed devices to the memory bus.

1.4 CONTROL OPTIONS

The Model 1 is available with two basic Control Panel options. The Standard Control Panel contains the normal control switches, data entry switches, Accumulator, Location Counter, Carry Bit, Enable Bit, WAIT, and Power Displays and a key-lock switch for power off and locked control.

The Auto Control Panel contains only a key-lock switch. The only control the operator can initiate is to turn on the power and restart the resident program, or reload a new program from a Teletype over the serial I/O port. This, in combination with the read only memory module, is an ideal, economical configuration for dedicated "Black Box" configurations.

1.5 PERIPHERALS

A complete line of peripherals is available with the Model 1. All system modules and device controllers that were designed for the Models 2, 3, 4, and 5 are plug and hardware compatible with the Model 1 Processor. These modules and device controllers are total designs, to handle not only conversions, buffering, timing, device commands and status, but also interrupts that can be enabled or disabled by the program. This broad line of peripherals includes controllers that INTERDATA developed and optimized for applications, such as the Digital Multiplexor for industrial monitoring and control. The following is an example of what is available for the Model 1. Chapter 8 covers some of these in more detail.

1.5.1 Digital Multiplexor

This provides a very economical set of modular building blocks to monitor 2048 and control 2048 lines with a single controller. Input or output modules of 128 lines can be intermixed for a total of 4096 lines per controller. The Digital Multiplexor utilizes a biased core technique to insure absolute DC isolation from the sense contact, an excellent common mode transient response and DC offset capability. These features make the unit particularly suitable for use in noisy environments.

1.5.2 Intertape System

The Intertape System provides a reliable and inexpensive substitute for paper tape input/output equipment. The transfer rate is 300 characters per second, with up to 250K bytes of storage per cassette, or 500K bytes total.

1.5.3 Mini Disc System

This is a rugged, reliable, and inexpensive mass storage system with 51,200 bytes of storage per disc. Up to two discs can be operated on each controller. Average access time is 8.5 milliseconds and the transfer rate is 60,000 bytes per second. The unit operates over a wide temperature range.

1.5.4 Magnetic Tapes

IBM compatible seven and nine track tape transports are available for the Model 1 with 25 ips speed and densities of 556 and 800 bpi.

1.5.5 Data Communications Equipment

Character buffered adapters are available with various options to Bell 103, 201, 202, 301 Data Sets and the 801 Automatic Dialer to provide a broad capability for remote applications with synchronous or asynchronous communications requirements.

1.5.6 Paper Tape Equipment

A 300 character per second Reader and 60 character per second Punch are offered for the Model 1, individually, or as a complete package. Fan-fold tape is featured as the software media.

1.5.7 Card Reader

A 200 cards per minute reader is provided for card oriented input systems.

1.5.8 System Modules

These modules provide the user with a group of general interfaces to reduce or eliminate design effort. As an example, modules are provided to handle eight-bit or sixteen-bit parallel input or output, manual data entry, and decimal indicators in four or eight decades.

1.5.9 Conversion Equipment

A line of analog-to-digital and digital-to-analog converters and analog multiplexors is available from eight-bit to twelve-bit, ±10 volt range.

1.5.10 Other Interfaces

Many other special interfaces are available which are not listed here or on the price list. Consult a sales office for further information.

1.6 SOFTWARE

The Model 1 is supported with a total programming package to ease the programmer's task. Basic software programs are the Assember, Editor, Debug, Loader, Unloader, and Diagnostic packages. INTERDATA also provides a Monitor that provides I/O support and operator services. INTERDATA has not forgotten its present users, however. For their convenience, a Model 1 Simulator and Assembler can be run on Models 3, 4 and 5.

1.6.1 Assembler

The Model 1 Assembler allows the user to write efficient and time-saving symbolic programs, which are translated by the Assembler into Model 1 machine language. The Assembler accepts a source deck or tape consisting of user-coded instructions, and outputs a source listing and a binary program object tape which can be loaded and executed by the Model 1 Loader.

The Assembler is designated to operate with a minimum of 4K bytes of memory. Assemblies are completed with two passes of the source medium through the resident assembler. The assembler can be

adapted to any configuration of I/O devices to optimize assembly speed. In addition to op-code mnemonics, the assembler has the capability of accepting up to six characters in user symbols. It can also decode instruction modifier mnemonics and has a flexible constant definition format.

1.6.2 Model 1 Text Editor

The Model 1 Text Editor is a support program for the Assembler used in the preparation and update of source program tapes. It is an interactive, Teletype-controlled program, but can use the high speed paper tape devices for input and output. The functions include adding, modifying, deleting, and copying of paper tape records. Tapes can be punched in assembly format or non-edited format for use as input for user programs. The Text Editor keyboard commands easily facilitate error correction and record or character manipulation.

1.6.3 Model 1 Debug

Model 1 Debug is an interactive hexadecimal debugging aid that contains many valuable program testing features. Among these are the display and modification of memory locations, and the accumulator. Also included is a breakpoint feature which allows the user to set, reset, and recognize breakpoints throughout his logic and enter his program at any point.

1.6.4 Model 1 Loader

The Model 1 Loader is a page-relocatable re-entrant program which loads absolute object tapes output from the Assembler. The Loader itself is loaded by a bootstrap loader in core. The Model 1 Loader performs a checksum on the data it loads and automatically transfers to the start of the user program.

1.6.5 Model 1 Diagnostics

The Model 1 Processor and Memory Tests are diagnostics which check the execution and validity of all Model 1 machine instructions including modifiers, and core memory with worst case patterns. Error messages result from any hardware discrepancies. The program repeats execution until halted or until an error is discovered. Test programs are also provided for the peripheral devices to validate the controller's and device's operational status.

1.6.6 Model 1 Monitor

The Model 1 Monitor provides the user with a convenient means of performing I/O operations with peripheral devices. The Model 1 Monitor is modularly constructed such that additional I/O drivers can be added to the system. Communication between the user and the Monitor is accomplished via a Teletypewriter.

1.6.7 Model 3, 4, 5/Model 1 Assembler

This program assembles Model 1 symbolic code into Model 1 object code as was described for the Model 1 Assembler. This software however runs on the INTERDATA Model 3, 4 and 5 with 8K bytes of memory for the convenience of these users.

1.6.8 Model 1 Simulator

This interactive software package enables the user to execute and test Model 1 object programs on an INTERDATA Model 3, 4 or 5 Processor with 8K bytes of memory. This program is also provided for the convenience of these users.

1.7 CUSTOMER SUPPORT

INTERDATA provides training courses on a year-round basis at several levels: system planning seminars, maintenance, and programming. If desired, special courses are presented at the customer's own facility.

1.7.1 Field Service

INTERDATA maintains Field Service Engineers across the country with headquarters in each of the regional sales offices. All of the INTERDATA field service personnel are factory trained and have experience with dozens of installations.

At the INTERDATA factory, back-up service is available, and a repair depot is maintained for logic boards and memory packages. In the field, maintenance personnel are equipped with complete sets of spares and with specialized diagnostic equipment. Working behind all this, is the factory Quality Control team which maintains exacting standards for production and final checkout. A customer's system is given hours of running tests and then, finally, "baked" in a heat chamber for a number of additional hours before the system is certified and shipped.

1.7.2 Training Center

INTERDATA takes customer training very seriously. A group of professional instructors is maintained for this purpose. The people on the training staff have all had prior experience in the training of computer personnel in military and civilian schools. Hundreds of customer engineers have already been successfully trained by this staff.

1.7.3 Systems Engineering

In many instances where the needs of the application go beyond standard hardware and software, customers have looked to INTERDATA for special assistance. An experienced team is organized to furnish this support, whether it be a contract for special hardware, or a special software package. The application team is made up of communications experts, programmers, analog specialists, and control specialists. They are an elite group in terms of capability and experience. Where a problem requires a special interface, the solution can often be built quickly and at nominal expense from off-the-shelf modules.

1.8 INTERCHANGE

INTERCHANGE, the INTERDATA users' group, is an active and growing association. By sharing software and special interfaces, customers gain a valuable "second level" of support.

CHAPTER 2
SYSTEM DESCRIPTION

2.1 GENERAL BLOCK DIAGRAM DESCRIPTION

The Model 1 Processor is an eight-bit byte-oriented digital computer useful in a wide variety of applications such as industrial control, data collection, communications, and general purpose computing. The Processor is modularly constructed using the latest in Medium Scale Integrated circuitry (MSI) and state of the art computer technology.

The Model 1 provides the user with more computing power per dollar than any machine available today. The Processor uses eight and sixteen-bit instructions for efficient coding and optimum core utilization. Many instructions have built-in test and skip capability for effective byte handling and loop control.

The Processor is designed around highly reliable 2048 by eight-bit byte core memory modules or plug compatible 2048 by eight-bit byte Read-Only-Memory modules. Both kinds of memory modules may be intermixed for up to 16,384 bytes of storage.

The memory system is organized into 64 pages containing 256 eight-bit bytes each. Two pages, Page Zero and the Current Page, are directly addressable by the Primary Instruction Word. All remaining pages are addressed indirectly. A powerful Auto Indexing feature enables the system to use up to 8,192 Index Registers located in core memory.

A Direct Memory Access Port is built into the Model 1 memory system. Up to four Direct Memory Access (DMA) Channels may be multiplexed to the port. The DMA Channels cycle-steal memory from the Processor at a maximum throughput rate of 1,000,000 bytes per second. Two kinds of DMA Channels are available; A Selector Channel for use with standard INTERDATA peripheral device controllers and a Universal Direct Memory Access Channel for applications involving custom channel design.

Input/Output operations between the Model 1 and the peripheral device controllers ordinarily occur over the Multiplexor Bus. Up to 256 device controllers may be connected to this bus. The Processor includes eight Input/Output Instructions plus Read Block and Write Block Instructions for high speed data transfer over the Multiplexor Bus. The Multiplexor Bus is fully hardware plug compatible with all peripheral device controllers developed and field proven on the existing line of INTERDATA Processors.

A Bit Serial I/O Port is standard on the Model 1 Processor. This provides a very economical method for interfacing to bit serial type devices such as Teletypewriters. The serial I/O port is interrupt-driven using the built-in 1.0 millisecond clock for controlling the port.

The Model 1 Processor features four hardware priority interrupt lines; four additional lines are available optionally. Each interrupt line is individually maskable and all interrupt lines are collectively enabled by a master Interrupt Enable Bit.

The Model 1 Processor is presented in this chapter in block diagram form. A general block diagram of the Processor is shown in Figure 2-1. The Processor's nine hardware registers are interconnected internally via eight-bit parallel data paths. The major elements of the Processor are described in the following paragraphs.

2.1.1 Program Status Word (PSW)

The Program Status Word is continually displayed on the standard Model 1 Control Panel. The program status word is a sixteen bit register. The fourteen least significant bits of this register form the Program Location Counter, LOC 2:15, which contains the memory location (address) of the next program instruction to be executed. Bit 1 corresponds to the master Interrupt Enable (E) Bit which, when set by the program, allows the Processor to respond to external interrupts. Bit 0 corresponds to the Carry (C) Bit, which is used during Arithmetic Instructions, Shift and Rotate Instructions, and certain Bit and Control operations.

2.1.2 Page Buffer Register (PBR)

The six-bit Page Buffer Register is used for temporary storage of the memory page address on indirect memory reference instructions.

2.1.3 Page Address Register (PAR)

The six-bit Page Address Register identifies one of 64 memory pages in the memory system.

2.1.4 Byte Address Register (BAR)

The eight-bit Byte Address Register identifies one of 256 bytes within a memory page. Note that the PAR together with the BAR form an effective fourteen-bit Memory Address Register for accessing up to 16,384 bytes of storage.

2.1.5 Memory Data Register (MDR)

The eight-bit Memory Data Register buffers data from the addressed memory location on all memory operations. It also contains the second operand on all Arithmetic and Logical Instructions.

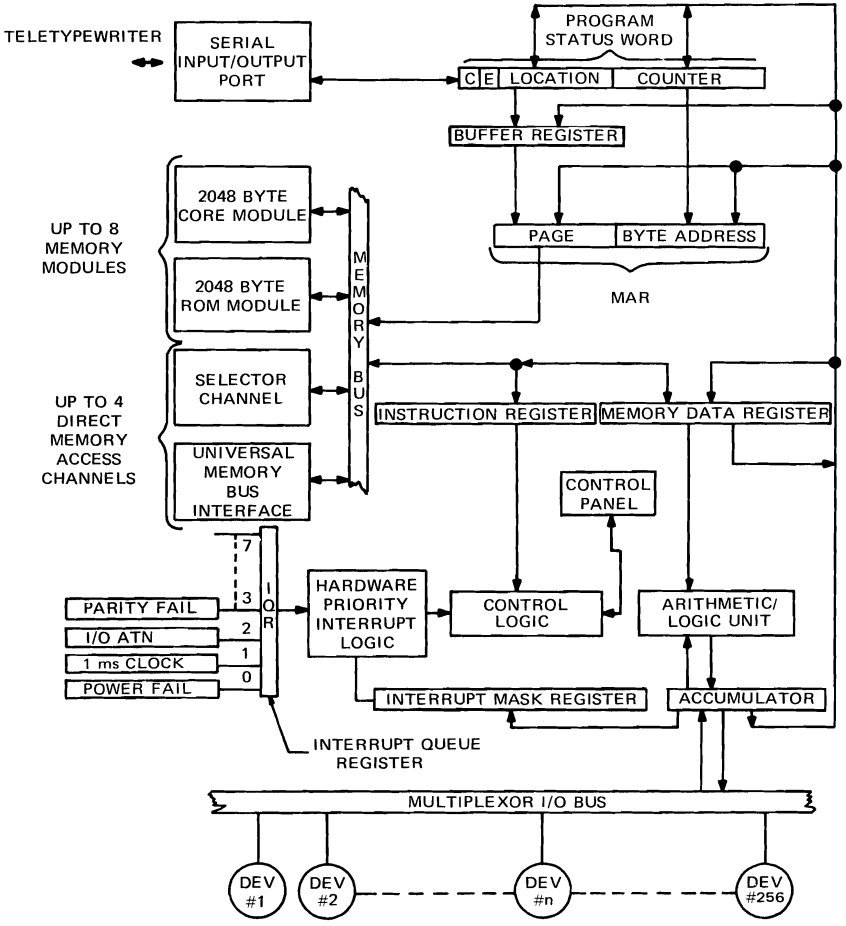


Figure 2-1. Model 1 Block Diagram

2.1.6 Instruction Register (IR)

The eight-bit Instruction Register contains the current Program Instruction being executed.

2.1.7 Accumulator Register (AR)

The eight-bit Accumulator contains the first operand on all Arithmetic and Logical Instructions and receives the result of such instructions. All bytes transferred over the Multiplexor Bus are transferred through AR.

2.1.8 Interrupt Queue Register (IQR)

The Interrupt Queue Register stores the event of an active condition on any of the four standard and the four optional external interrupt lines.

2.1.9 Interrupt Mask Register (IMR)

The Interrupt Mask Register selectively enables individual interrupt lines under program control. Bits 0 to 7 of the mask register refer to interrupt lines 0 to 7 respectively. Line 0 is top in priority, line 7 is bottom in priority.

2.1.10 Serial Input/Output Port

The Serial Input/Output Port is an economical method of interfacing to a bit serial data stream such as that associated with a Teletypewriter. Data is transferred from/to the port through the Carry Bit under the control of the program. The Serial I/O Port may be interrupt-driven and therefore does not require 100% dedicated Processor support.

2.1.11 Multiplexor I/O Bus

The Multiplexor Bus is a byte-oriented Input/Output Bus which can interface up to 256 peripheral device controllers. Data transfer over the Multiplexor Bus is done on a request/response basis, through the Accumulator. The Model 1 Multiplexor Bus is fully hardware plug compatible with all INTERDATA Peripherals.

2.1.12 Memory Bus

The Processor communicates with the Model 1 memory system over the common System Memory Bus. Up to four Direct Memory Access Channels (either Selector Channels or Universal Direct Memory Access Channels) may cycle steal from the Processor over the System Memory Bus. When a Direct Memory Access Channel is actively using memory, the Processor is stopped. The maximum data rate through the Direct Memory Access Channel is 1,000,000 bytes per

second. Control for serving DMA Channel memory requests is built into the Processor.

2.1.13 Memory Modules

Three kinds of memory modules may be installed into the Model 1 Memory System:

- Core Memory Modules - 2048 by eight-bit, 1.0 microsecond, random access modules, complete on a single standard sized INTERDATA circuit board.
- Parity Core Memory Modules - 2048 by nine bit, 1.0 usec, random access modules, complete on a single standard sized INTERDATA circuit board, where the ninth bit corresponds to the parity bit.
- Read-Only-Memory Modules - 2048 by eight-bit, 1.0 microsecond, random access Read-only-Modules, fully plug compatible with the Model 1 core memory modules above.

All three kinds of memory modules may be intermixed in any way (as long as the first module is a core memory) in a Model 1 system for a total of 16,384 bytes of storage (eight modules).

2.2 INTERRUPTS

2.2.1 Introduction

In very general terms, an interrupt is defined as an event which, when recognized by the Processor, causes the Processor to stop executing the current program and to begin executing a new program. This is done by saving the current PROGRAM STATUS WORD at a predetermined fixed memory address and then causing the PROGRAM STATUS WORD to be set to the starting location of the new program. When this is done, the Processor is said to have honored the interrupt. The new program will "service" the interrupt. When the interrupt has been serviced, the old program can be continued by restoring the PROGRAM STATUS WORD to the value previously saved at the fixed address. Note that the PSW includes the Carry Bit, the Interrupt Enable Bit, and the Location Counter.

Interrupts can be classified into one of two broad catagories: 1) Internal Interrupts are those events which are generated within the Processor; and 2) External Interrupts are those events which occur outside the Processor.

The Model 1 Processor has three sources of internal interrupts: 1) the Power Fail Interrupt interrupts the Processor if the CLEAR Switch on the Control Panel is depressed, the POWER Switch on the

Control Panel is turned OFF, or if a Primary Power Failure is detected by the Power Fail Safe option, 2) the one millisecond clock can interrupt the Processor once every millisecond, and 3) the Parity Fail Interrupt interrupts the Processor if a parity failure occurs.

External Interrupts can come from a wide variety of sources. One of these sources is the ATTENTION (ATN) line associated with the standard INTERDATA Multiplexor Bus (I/O Bus.) See Chapter 5 covering the I/O system.

The ATN Interrupt is generated by most standard INTERDATA peripheral device controllers when conditions in the controller require that the Processor be interrupted. The other Interrupt Lines may be connected to any equipment source the system designer chooses (i.e. relay contacts, switch contacts, signals from customer designed circuits, or even standard INTERDATA equipment whose interrupt line (ATN) is disconnected from the Multiplexor Bus and reconnected to a unique Interrupt Line).

An Interrupt Line is defined as a physical point which, when activated by some interrupt source, causes the Processor to save the old PSW at an address unique to that line. Thus, the event of an interrupt on one line causes the Processor to be directed to a new program different from that corresponding to an interrupt on a second interrupt line. The Model 1 Processor has four interrupt lines as standard equipment. Four additional interrupt lines may be added as an option.

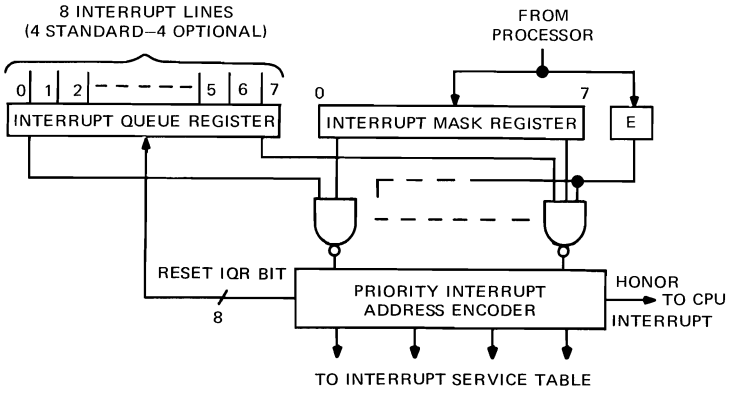
A priority is assigned to each interrupt line in the Processor. The notion of priority implies that the highest priority line takes precedence over all lower priority lines. Thus, the Processor will not recognize a low priority interrupt line so long as a higher priority line is active, and enabled.

2.2.2 Block Diagram

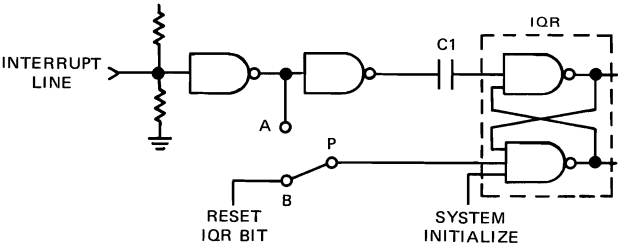
The block diagram in Figure 2-2 illustrates the hardware associated with the Interrupt Lines in the Model 1 Processor. Interrupt Line 0 is highest in priority, and Interrupt Line 7 is lowest in priority. The following paragraphs describe the major functions shown on Figure 2-2.

2.2.2.1 Interrupt Queue Register (IQR)

The IQR stores the event of an interrupt on any of the eight interrupt lines. The active input to the IQR may be either a negative-going edge or a low level (all inputs are standard DTL logic levels). Edge sensitive inputs are useful when the driving source supplies a short pulse or when the source is a "bouncy" metallic contact. Level



NOTES:
- INTERRUPT LINE "0" IS TOP PRIORITY; INTERRUPT LINE "7" IS LOWEST PRIORITY.
- THE HIGHEST PRIORITY "IQR" BIT WHICH IS ENABLED BY THE CORRESPONDING BIT IN THE "IMR" IS HONORED FIRST.
- "E" IS RESET WHEN ANY INTERRUPT LINE IS HONORED.
- INPUTS TO THE INTERRUPT QUEUE REGISTER MAY BE EDGE OR LEVEL SENSITIVE: THE SKETCH BELOW SHOWS A TYPICAL BIT IN THE "IQR" ARRANGED FOR NEGATIVE EDGE SENSITIVITY. THE "IQR" BIT IS SET BY A NEGATIVE TRANSITION ON THE INTERRUPT LINE. IT IS AUTOMATICALLY RESET WHEN THE PROCESSOR HONORS THAT LINE.



- ANY BIT IN THE "IQR" MAY BE CONVERTED TO A LEVEL SENSITIVE BIT BY SHORTING CAPACITOR C1, AND BY CONNECTING POINT P TO POINT A.
- CONVERSION FROM EDGE TO LEVEL SENSITIVE INPUTS IS EASILY DONE BY THE USER.
- INTERRUPT LINE 0, RESERVED FOR THE POWER FAIL, CLEAR, POWER OFF INTERRUPT IS NOT MASKED BY THE "E" BIT, BUT IT IS MASKABLE BY BIT 0 OF THE "IMR".

Figure 2-2. Model 1 Interrupt Lines and Associated Hardware

sensitive inputs are useful when more than one source OR ties onto a given line.

2.2.2.2 Interrupt Mask Register (IMR)

The IMR contains a mask which individually enables or disables each bit in the IQR. The IMR is loaded from the accumulator under program command.

2.2.2.3 Interrupt Enable (E) Bit

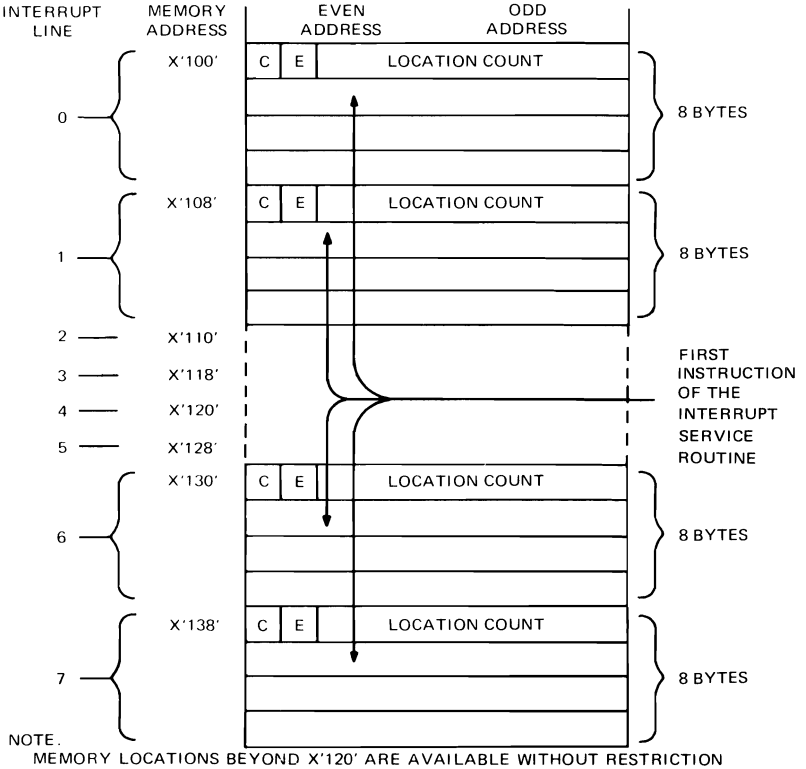
The E Bit is a master interrupt Enable/Disable Bit. When E is reset, the Processor ignores all interrupts (except Line 0 which is the Power Fail, Power ON/OFF, and CLEAR line) regardless of the state of IQR or IMR. When E is set, the Processor will respond to the highest

priority interrupt line which is both active and enabled by its corresponding Mask Bit. When the Processor honors an interrupt, the current PSW is saved, the E Bit is unconditionally reset, and remains so until the program commands it set.

Note: The E Bit does not affect Interrupt Line 0 which is reserved for the Power Fail Interrupt. Bit 0 of IMR does affect the Power Fail Interrupt.

2.2.2.4 Interrupt Service Table

Associated with each interrupt line is an eight-byte set of consecutive core memory locations in Page One of the memory system (See Figure 2-3). Accordingly, 32 bytes of core are reserved for a Processor with four interrupt lines, and 64 bytes are reserved for a Processor with eight interrupt lines. These reserved core addresses are referred to as the Interrupt Service Table.



NOTE.
MEMORY LOCATIONS BEYOND X'120' ARE AVAILABLE WITHOUT RESTRICTION IF THE PROCESSOR DOES NOT HAVE THE OPTIONAL 4 INTERRUPT LINES (4:7)

– THE FIRST INSTRUCTION OF THE NEW PROGRAM CORRESPONDING TO INTERRUPT LINE 0 WOULD BE AT LOCATION X'102'

Figure 2-3. Interrupt Service Table

The Interrupt Service Table is subdivided into four or eight sets of eight bytes of storage each; one set for each interrupt line. As

indicated in Figure 2-3, memory addresses X'100' through X'107' are reserved for Interrupt Line 0; and so on.

When the Processor honors an interrupt line, it stores the old Program Status Word (the old LOCATION COUNT plus the Carry Bit and the Enable Bit) in the first two bytes of the eight-byte set associated with that interrupt line. The Processor then begins executing program instructions starting at the third byte in that eight-byte set. Service programs longer than six bytes must branch to some memory area outside the Service Table region.

When the new program has finished servicing the interrupt, it performs a program Branch Instruction, indirectly through the first two bytes of the eight-byte set, which restores the old Program Status Word.

2.2.3 General Notes on Interrupts

The four standard interrupt lines plus the four optional interrupt lines in the Processor may be connected, at the user's option, in a wide variety of ways. However, for purposes of standardization, the first three interrupt lines are factory connected as follows:

Interrupt Line 0 - Connected to the Control Panel POWER OFF and CLEAR Switches and to the optional Power Fail Detect circuit. Note that these internal interrupts require program support as described in Chapter 10 of this manual. This line is not masked by the Enable (E) Bit, it is masked by Bit 0 in the IMR.

Interrupt Line 1 - Connected to the Processor's built-in one millisecond clock.

Interrupt Line 2 - Connected to the standard Multiplexor Bus Attention (ATN) Line. The ATN Line is a common interrupt line for up to 256 device controllers on the Multiplexor Bus. The program responds to an interrupt on this line by executing an Acknowledge Interrupt Instruction. The interrupting device controller's device number is placed in the accumulator as a result of the Acknowledge Interrupt Instruction.

Interrupt Line 3 - Reserved for Parity Option if the System has that option. Otherwise it is available to the user.

Interrupt Line 4

. These lines are available to the user.
.

Interrupt Line 7 -

The user may conveniently interchange the first four interrupt lines with any other lines to re-establish priorities if he chooses to do so. He must be aware that Line Zero is not under control of the Interrupt Enable (E) Bit.

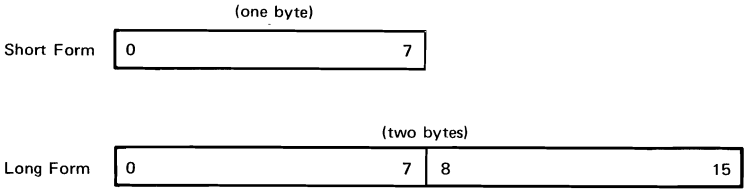
CHAPTER 3

INSTRUCTION FORMATS AND ADDRESSING MODES

3.1 INTRODUCTION

A program is a set of instructions which directs the Processor to perform some task. Ordinarily program instructions are stored in sequential memory locations. The Processor reads each instruction from the memory, executes that instruction, then reads the next instruction in sequence, executes it, and so on.

Model 1 Instructions fall into two broad catagories designated Short Form Instructions and Long Form Instructions. A Short Form Instruction is eight bits (one byte) long and is stored in one memory location. A Long Form Instruction is sixteen bits (two bytes) long and is stored in two adjacent memory locations.



The Location Counter contains the memory location of the next program instruction to be executed. The Location Counter is incremented each time an instruction is read from memory. It is incremented once for Short Form Instructions, it is incremented twice for Long Form Instructions.

Nearly all instructions have built-in test and skip capability allowing the program to skip over the next instruction in sequence when certain specified conditions are true.

If the instruction is a Short Form Instruction, the Skip conditions are tested before the instruction is executed. If the instruction is a Long Form Instruction, the Skip conditions are tested after the instruction is executed.

In any case, the next instruction could be either a Short or a Long Form Instruction. When a skip is taken, the Processor always assumes that the next instruction is in the Long Form and accordingly, it increments the Location Counter two additional times. Figure 3-1 illustrates this action.

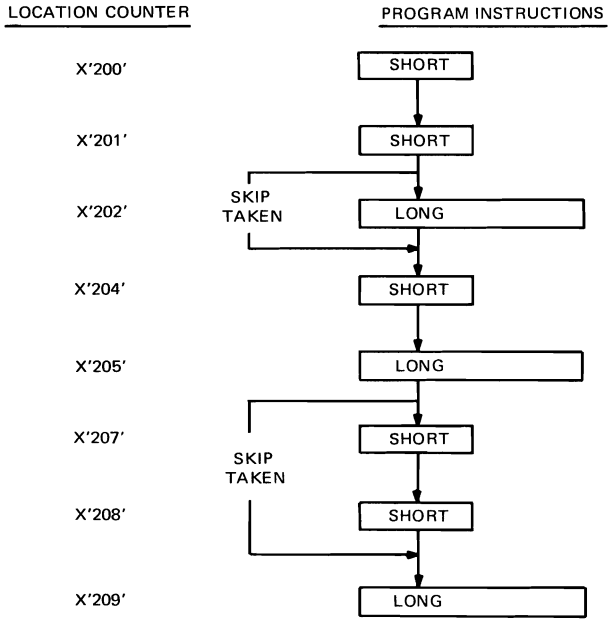


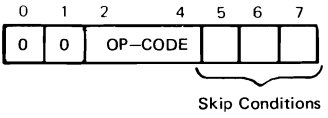
Figure 3-1. Skip Examples

3.2 INSTRUCTION FORMATS

Six Instruction Formats are defined for the Model 1 Processor. Two Instruction Formats are Short Form eight-bit instructions. The remaining four Instruction Formats are Long Form sixteen-bit instructions. The individual instructions are described in detail in Chapter 4.

3.2.1 A Register/Carry Format

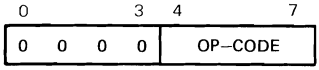
These instructions are eight-bit Short Form Instructions which operate on either the Accumulator or on the Carry Bit, according to the Operation Code (Op-Code).



Instructions in this format can specify skip conditions which are tested before the instruction is executed.

3.2.2 Input/Output Format

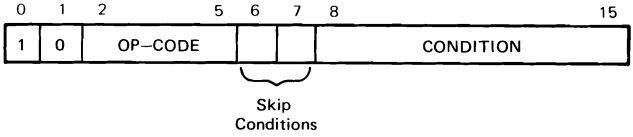
These instructions are eight-bit Short Form Instructions which transfer eight-bit bytes between the Multiplexor Bus and the Accumulator.



Communication and data transfers over the Multiplexor Bus are request/response in nature. The Op-Code defines the direction of data flow. There are four basic Output I/O Instructions and four basic Input I/O Instructions in this format. The final instruction in this format is the Pulsed I/O Instruction which provides a convenient and economical method of transferring eight-bit bytes of data to/from the Accumulator and custom designed circuits. Instructions in the Input/Output format cannot specify skip conditions although two instructions include an implied test and skip.

3.2.3 Command Format

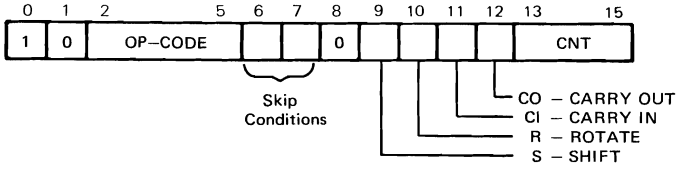
These are sixteen-bit Long Form Instructions which allow the system programmer to set up a wide variety of internal Processor conditions.



The eight-bit CONDITION field identifies the particular combination of Command Functions to be performed. Instructions in this format can specify two Skip conditions which are tested after the instruction is executed.

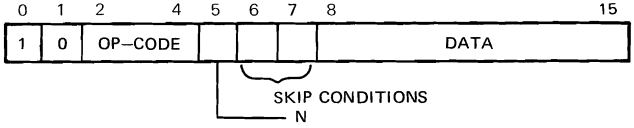
3.2.4 Shift/Rotate Instruction Format

These instructions are sixteen-bit, Long Form Instructions which cause the Accumulator to be shifted or rotated to the right the number of bit positions specified in the CNT field. These instructions include full Carry In and Carry Out control and may specify two Skip conditions which are tested after the instructions is executed.



3.2.5 Immediate Format

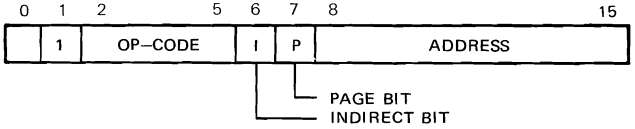
These are sixteen-bit Long Form Instructions which operate on the Accumulator with an eight-bit data byte which is part of the Instruction Word.



Immediate operations include ADD, SUBTRACT, AND, OR, EXCLUSIVE-OR, and LOAD. The result of the operation is placed in the Accumulator unless the N Bit is set. In this case, the Accumulator is not modified, but the instruction is otherwise executed, the results are tested, and a Skip is taken if any specified condition is true.

3.2.6 Memory Reference Format

These are sixteen-bit Long Form Instructions which generally operate on the Accumulator with an eight-bit data byte which is contained in memory.

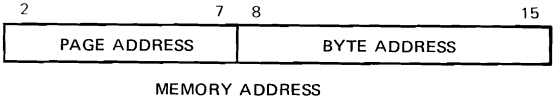


The Instruction Word contains an eight-bit Address field plus two bits to identify one of four Addressing Modes; Direct in Page Zero, Direct in Current Page, Indirect through Page Zero, and Indirect through the Current Page. Indirect memory reference instructions may specify a powerful Auto-Indexing feature using any adjacent pair of bytes in core as an index register.

3.3 ADDRESSING MODES

A fully expanded Model 1 Memory System includes 16,384 eight-bit bytes of storage. The memory is divided into 64 equal sized segments called Pages. Each Page contains 256 bytes of storage. See Figure 3-2.

Any unique eight-bit memory location is defined by a fourteen-bit memory address; six bits for the Page Address and eight bits for the Byte Address within that Page.



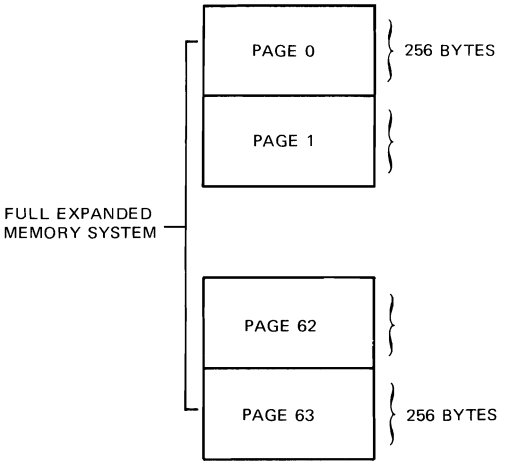
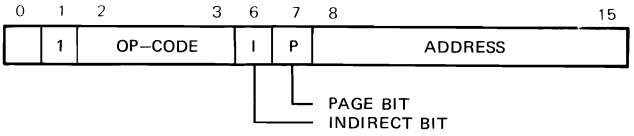


Figure 3.2. Pages in the Memory System

3.3.1 Direct Addressing Mode

Memory Reference Instructions have the following general form.



When the I Bit (Bit 6) is reset, the Addressing Mode is designated DIRECT. The eight-bit Address Field corresponds directly to the Byte Address of the instruction's operand. When the Page Bit (Bit 7) is reset, the Addressing Mode is DIRECT in PAGE ZERO. When the Page Bit is set, the Addressing Mode is DIRECT in the CURRENT PAGE. Thus, Memory Reference Instructions can directly address up to 512 bytes (256 bytes each in PAGE ZERO and the CURRENT PAGE). See Figure 3-3.

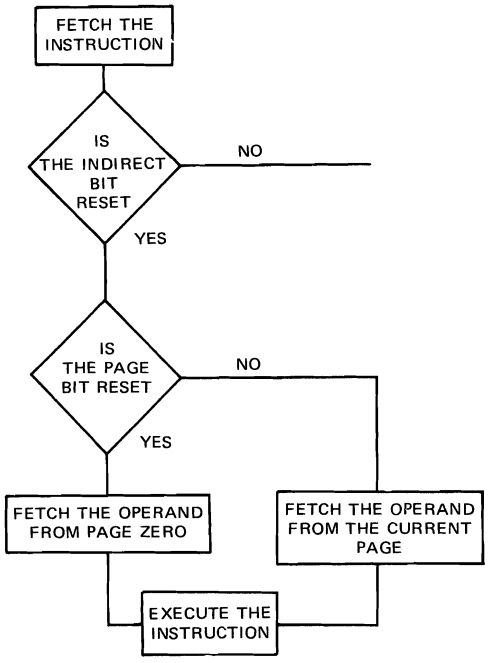
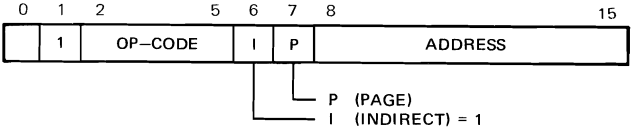


Figure 3-3. Flow Chart - Direct Addressing Mode

3.3.2 Indirect Addressing Mode

Memory Reference Instructions have the following general form:



When the I Bit (Bit 6) is set, the Addressing Mode is designated INDIRECT. The eight-bit Address Field identifies the first byte of a two byte pointer which contains the effective address of the operand. The pointer is any two consecutive byte pair in either PAGE ZERO or the CURRENT PAGE as indicated by the P Bit (Bit 7). When the P Bit is reset, the Pointer lies in PAGE ZERO. When the P Bit is set, the Pointer lies in the CURRENT PAGE. Indirect Memory Reference Instructions may operate on data anywhere in core. See Figure 3-4.

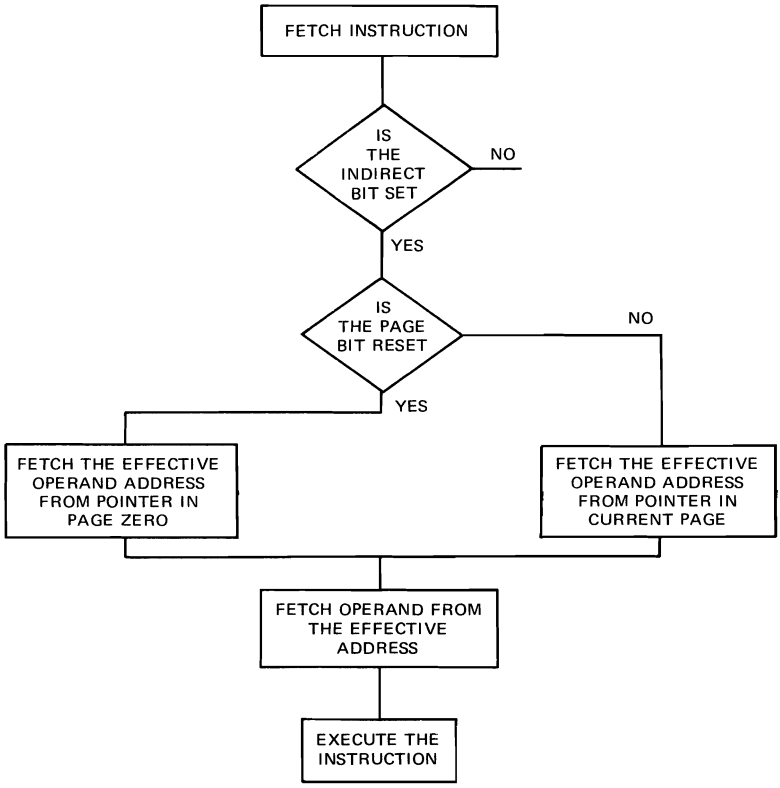
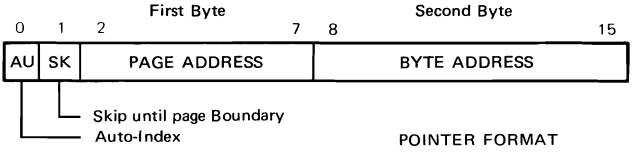


Figure 3-4. Flow Chart - Indirect Addressing Mode

3.3.3 Auto-Indexing

Auto-Indexing is a very powerful feature which is useful for efficient program looping and loop control. Indirect Memory Reference Instructions may get the Auto-Index function when the AU Bit in the pointer is set. When the AU Bit is set, the Processor increments the Byte Address of the Pointer, before it is used, every time the Indirect Instruction is executed. See Figure 3-5.

Note: Only the Byte Address of the pointer is incremented.



Note: SK = 1 is ignored if AU = 0

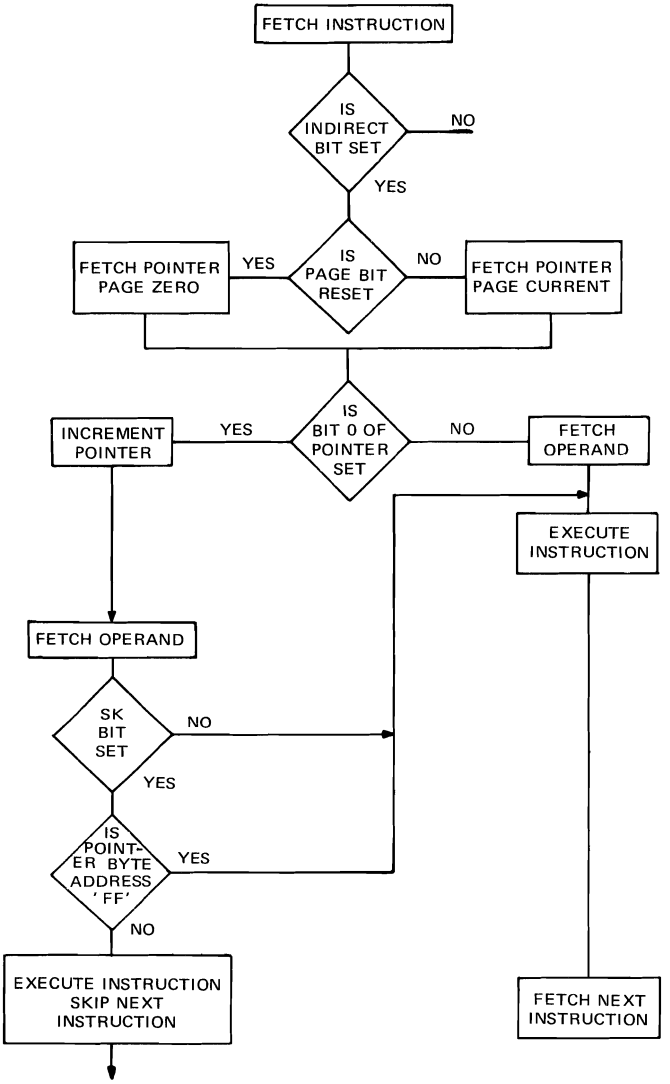


Figure 3-5. Auto Indexing

Associated with the Auto-Index function is a conditional Skip on Page Boundary feature which is specified by setting the SK Bit. If the SK bit is set, the Processor skips after executing the instruction unless the resultant Byte Address in the Pointer, after incementing, equals all ONES ('FF').

3.4 ADDRESSING MODE SUMMARY

The flow chart in Figure 3-6 summarizes the types of Addressing Modes and Auto-Indexing in the Model 1.

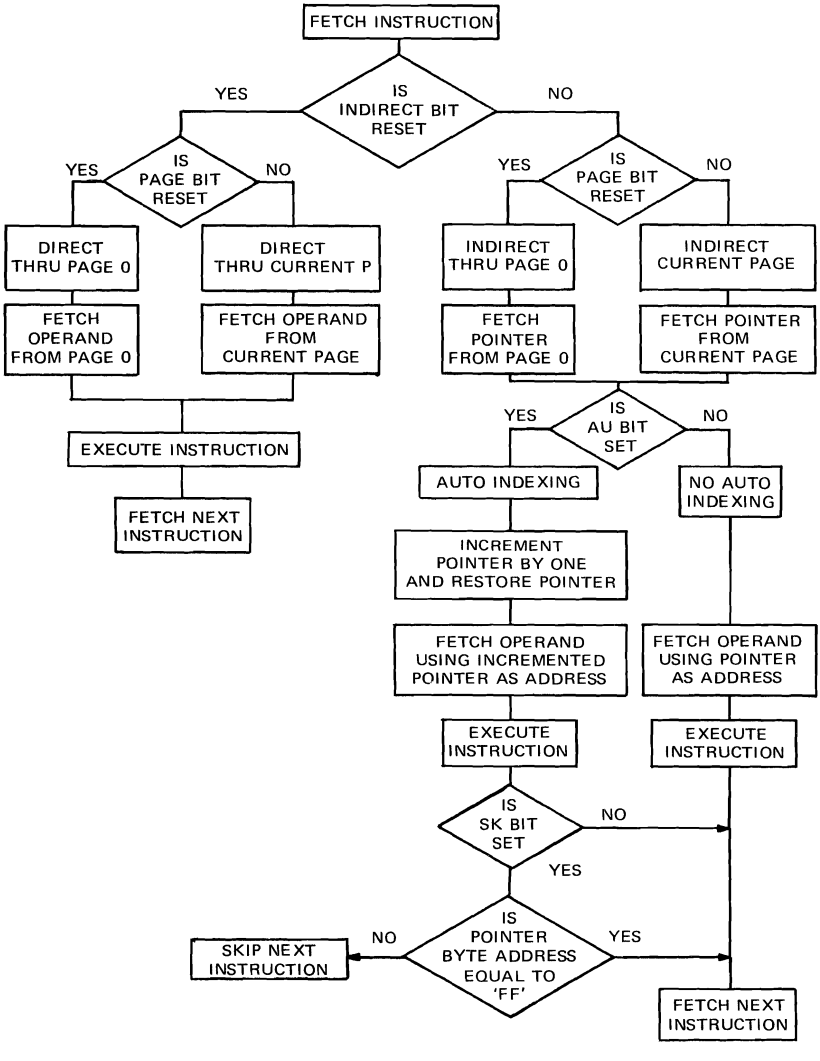


Figure 3-6. Flow Chart - Addressing Modes/Auto Indexing

CHAPTER 4
INSTRUCTION REPERTOIRE

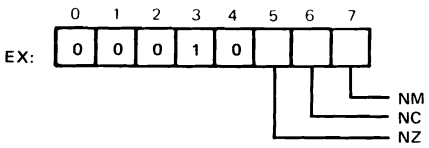
4.1 INTRODUCTION

The Instruction Repertoire has been grouped by class in this chapter. For each class of instructions a general description of the class is given, followed by an explanation of each instruction belonging to that class. For individual instruction descriptions the following format is used.

1) The name of the instruction followed by its mnemonic op-code.

EX: ADD ONE TO A AO

2) The instruction bit format.



3) A description of the instruction operation

4) An explanation of resulting effects to the accumulator or carry.

5) A diagramatic representation of the instruction

EX: Accumulator ← Accumulator + One

6) Programming notes or examples to provide additional pertinent or clarifying information.

4.2 A REGISTER/CARRY INSTRUCTIONS

The A Register/Carry Instructions are Short Form eight-bit Instructions which operate on either the Accumulator or the Carry Bit. Three valid modifiers may specify conditions on A Register operations:

NZ - Skip if A is not ZERO
NC - Skip if Carry is not set
NM - Skip if A is not minus

Only one valid modifier may be specified on Carry Bit operations.

NC - Skip if Carry is not set

Skip conditions are tested before the A Register/Carry Instruction is performed. Thus, the skip conditions tested reflect the result of the previous instruction (or series of instructions) See Figure 4-1.

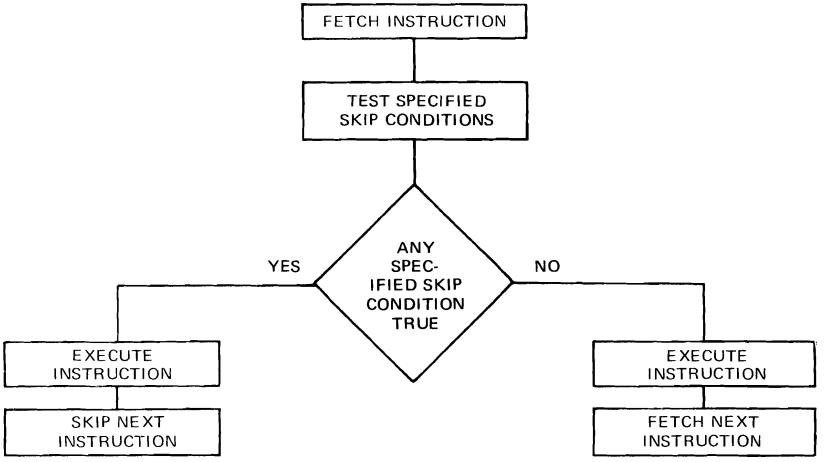
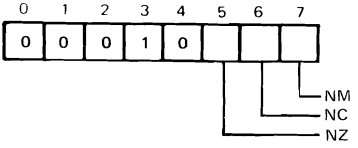


Figure 4-1. Flow Chart - A Register/Carry Instructions

4.2.1 Add One to A AO

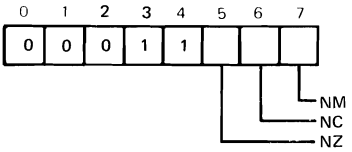


The specified skip conditions are tested, the value ONE is added to the Accumulator, and the next program instruction is skipped if any specified skip condition was true.

Carry Bit will reflect the result of the Carry Out from Bit 0 of the Accumulator.

Accumulator ← Accumulator + ONE

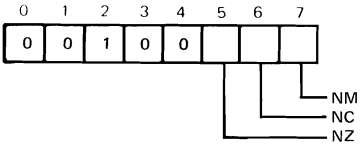
4.2.2 TEST and SKIP TS



The specified Skip conditions are tested and the next program instruction is skipped if any specified Skip condition is true.

The Carry Bit is not changed. The Accumulator is not changed.

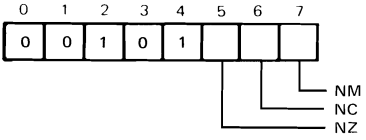
4.2.3 Complement A CA



The specified Skip conditions are tested, the A Register is replaced by the ONE's complement of A, and the next program instruction is skipped if any of the specified Skip conditions were true.

The Carry Bit is not changed.
Accumulator ← ONE's Complement of Accumulator.

4.2.4 Add A To A With Carry Out AA



The specified Skip conditions are tested, the A Register is added to itself, and the next program instruction is skipped if any of the specified Skip conditions were true.

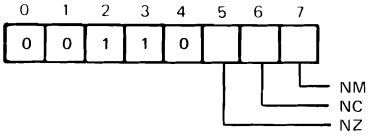
NOTE

This instruction is effectively a Shift A left one position, with Carry Out.

Carry Bit will reflect the result of the Carry Out from Bit 0 of the Accumulator.

Accumulator ← Accumulator + Accumulator

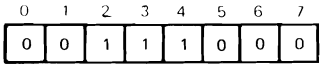
4.2.5 Clear Accumulator CLR



The Skip conditions are tested, the A Register is reset to all ZEROs, and the next instruction is skipped if any specified Skip condition was true.

The Carry Bit is not changed. Accumulator ← ZEROs

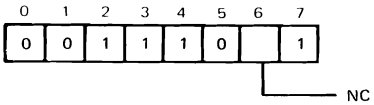
4.2.6 No Operation NOP



Nothing is altered or executed by this instruction.

Carry Bit is not changed. Accumulator is not changed.

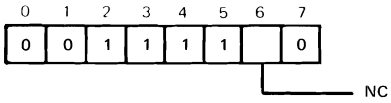
4.2.7 Set Carry SC



The specified Skip condition is tested, the Carry Flag is unconditionally set, and the next program instruction is skipped if the specified condition was true.

Carry Bit ← ONE. Accumulator is not changed.

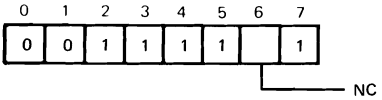
4.2.8 Reset Carry RC



The specified Skip condition is tested, the Carry Flag is unconditionally reset, and the next program instruction is skipped if the specified condition was true.

Carry Bit ← ZERO. Accumulator is not changed.

4.2.9 Complement Carry CC



The specified Skip condition is tested, the Carry Flag is unconditionally complemented, and the next program instruction is skipped if the specified condition was true.

Carry Bit ← Complemented Carry Bit. Accumulator is not changed.

4.3 SHIFT/ROTATE INSTRUCTIONS

The Shift/Rotate Instructions are Long Form sixteen-bit instructions which logically shift or logically rotate the Accumulator the specified number of bit positions with full Carry In and Carry Out control. The direction of the Shift or Rotate is always right. The number of bit positions specified by these instructions may lie anywhere in the range from one to eight.

Two Skip conditions may be specified on Shift/Rotate Instructions:

NC - Skip if the Carry Bit result is not set.
NM - Skip if the Accumulator result is not negative.

The Skip conditions are tested at the end of the instruction execution. If any specified Skip condition is true, the next program instruction is skipped. Otherwise the next program instruction is executed. See Figure 4-2.

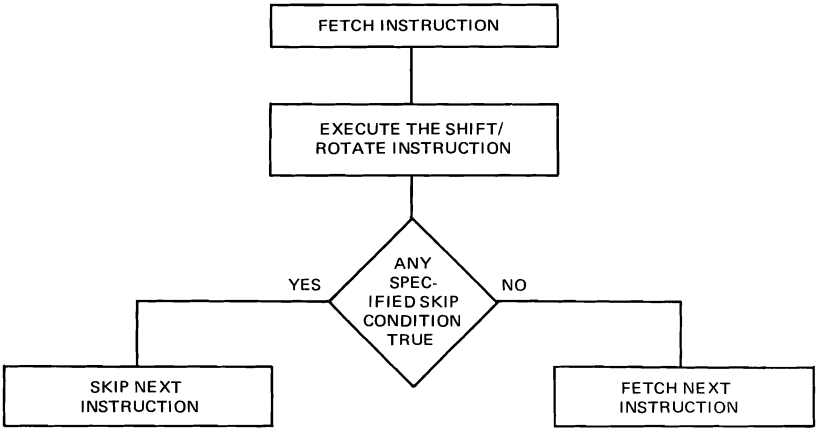
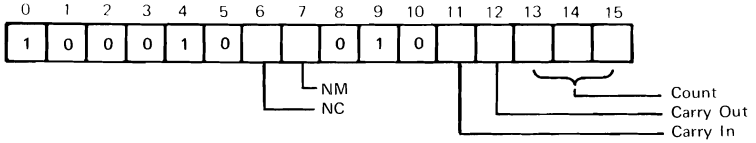


Figure 4-2. Shift/Rotate Instruction

4.3.1 Shift SH



The Accumulator is logically shifted right the number of bit positions specified by the Count Field. The Count Field must specify the ONEs complement of the number of shift positions desired, from one to eight. (The ONEs complement operation is done by the Model 1 Assembler.) If Carry Out (CO) is specified, the bit shifted out of A7 is placed in the Carry Bit. If Carry In (CI) is specified, the Carry Bit is shifted into A0. If any specified Skip condition is true at the end of the instruction, the next program instruction is skipped. See Figure 4-3.

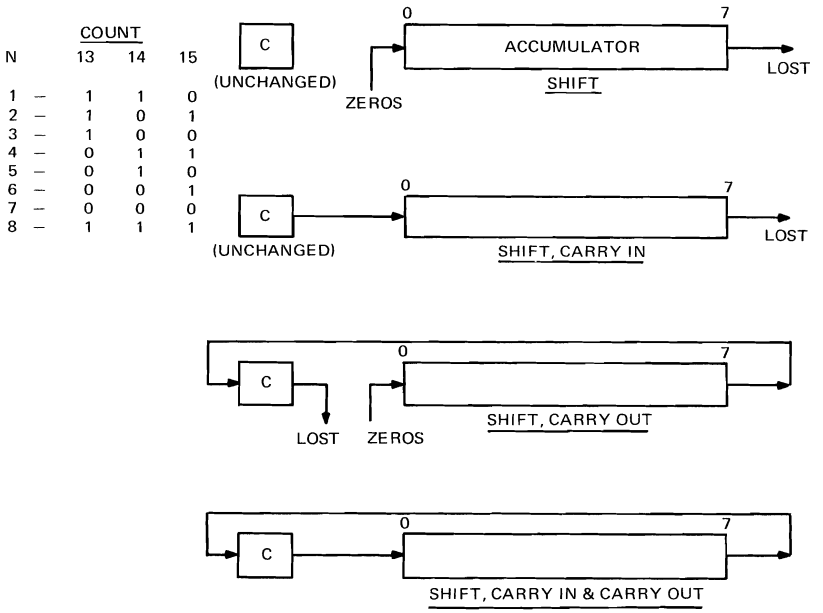
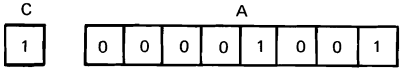


Figure 4-3. Example of Shift Instruction

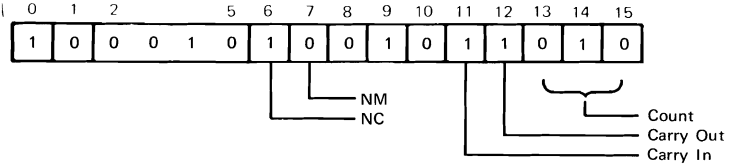
Example: If



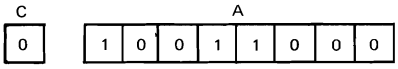
and we execute at location 202

SH 5, CI, CO, NC

The instruction takes the form (One's Comp of 5 = 010)



the result is,

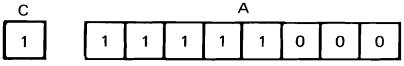


and the program skips from address 204 to 206.

Example: If above were

202 SH 5, CI, NC

the result is

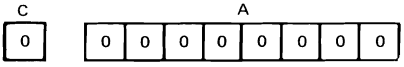


and the program does not skip.

If above were

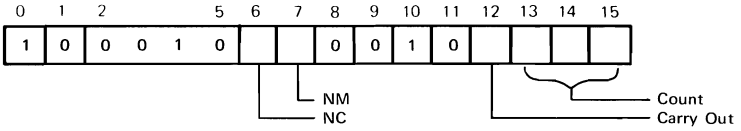
202 SH 5, CO, NC

the result is



and the program does skip.

4.3.2 Rotate RT



The Accumulator is logically rotated right the number of bit positions indicated by the Count Field. The Count Field must specify the ONEs complement of the number of bit positions, from one to eight. The bit rotated out of A7 is placed in A0. If Carry Out (CO) is specified, the bit rotated out of A7 is placed in both the Carry Bit and in A0. If any of the specified Skip conditions are true at the end of the instruction, the next program instruction is skipped. See Figure 4-4.

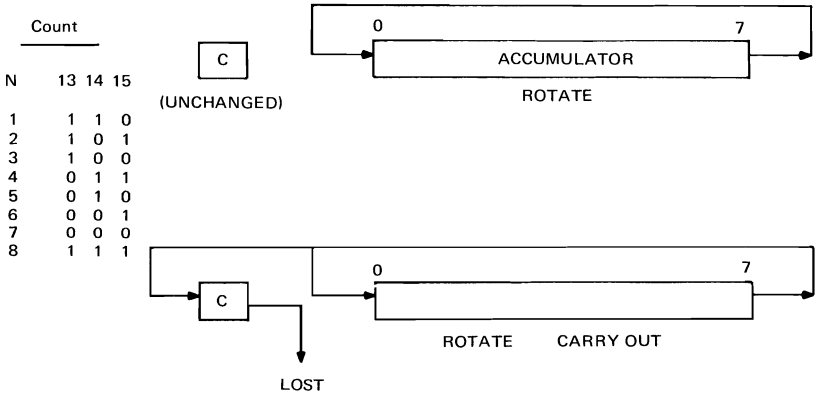
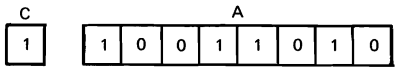


Figure 4-4. Example of Rotate Instruction

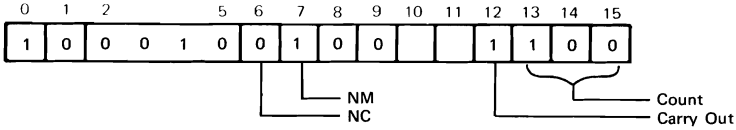
Example: If



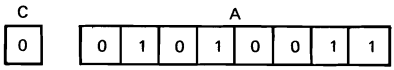
and we execute at location 202

RT 3, CO, NM

the instruction takes the form One's Comp of 3 = 100



the result is

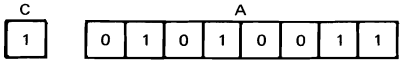


and the instruction at 204 is skipped.

Example: If above were

202 RT 3, NM

the result is



and the instruction at 204 is skipped.

4.4 INPUT/OUTPUT INSTRUCTIONS

Input/Output (I/O) Instructions fall into three categories; Normal I/O Instructions, Pulsed I/O Instructions, and Block I/O Instructions. Normal I/O and Pulsed I/O Instructions are Short Form (eight-bit) Instructions and all data transferred resides in the Accumulator. Block I/O Instructions are Long Form (sixteen-bit) Memory Reference Instructions and all data transferred by them resides in a Memory Page.

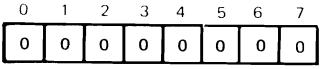
The Normal I/O Instructions operate with any standard INTERDATA device controller. The Block I/O Instructions operate with all INTERDATA Controllers capable of operating in a Block Mode. The Pulsed I/O Instruction is intended for custom designs and is not generally appropriate for standard INTERDATA Controllers.

Normal I/O and Pulsed I/O Instructions transfer data to/from device controllers and the Accumulator. Block I/O Instructions transfer data to/from device controllers and memory.

The Pulsed I/O Instruction never changes the Carry Bit. Ordinarily, the Normal and Block I/O Instructions do not affect the Carry Bit either. There is an exception to this however. If the addressed device controller fails to respond within approximately 35 microseconds to any Normal or Block I/O Instruction, the instruction will be aborted, the Carry Bit is unconditionally set, and the next instruction is executed.

Detailed descriptions of the Multiplexor Bus are given in Chapter 5.

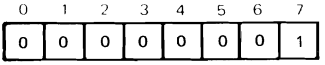
4.4.1 Address ADR



The device controller whose address corresponds to the content of the Accumulator is addressed. The Address flip-flop of all other device controllers is reset.

The Carry Bit is not changed unless the device does not respond within the time out period. The Accumulator is not changed.
Device Controller◄——Accumulator (Address)

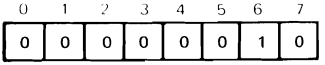
4.4.2 Output Command OC



The currently addressed device controller receives an eight-bit command byte from the Accumulator.

The Carry Bit is not changed unless the device does not respond within the time-out period. The Accumulator is not changed.
Addressed Device Controller◄——Accumulator (Command Byte)

4.4.3 Write Data WD



The currently addressed device controller receives an eight-bit data byte from the Accumulator.

The Carry Bit is not changed unless the device does not respond.
The Accumulator is not changed.
Addressed Device◄——Accumulator (Data Byte)

4.4.4 Write Data and Skip WDS

0	1	2	3	4	5	6	7
0	0	0	0	0	0	1	1

The status of the currently addressed device controller is sensed and tested.

If the transfer can be made:

When the device is ready, an eight-bit data byte is transferred to the device from the Accumulator. The next program instruction is skipped on the successful execution of this instruction. See Figure 4-5.

The Carry Bit is not changed unless the device does not respond within the time-out period. The Accumulator is not changed.
Addressed Device ← Accumulator (Data Byte)

If the transfer cannot be made:

If any of bits 5, 6, or 7, of the device status is true, the instruction is aborted, the offending status byte is loaded to the Accumulator, and the next program instruction is not skipped.

The Carry Bit is not changed unless the device does not respond.
The Accumulator ← Status Byte.

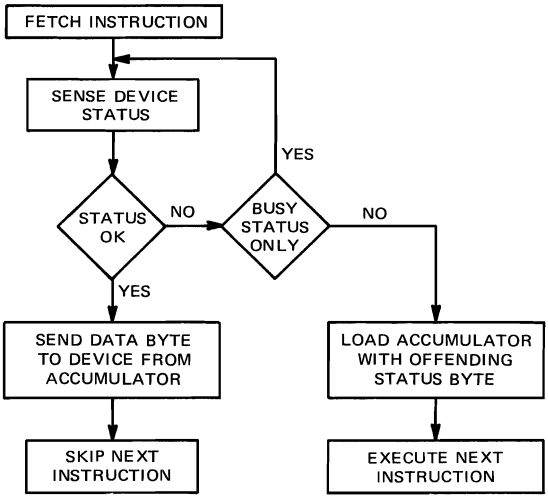
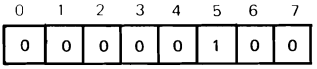


Figure 4-5. Write Data and Skip Flow Chart

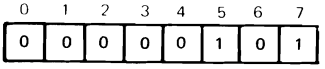
4.4.5 Acknowledge AK



This instruction is issued in response to an I/O Interrupt, ATN, over the Multiplexor Bus. The device address corresponding to the interrupting device is placed in the Accumulator, and ATN for that device is reset.

The Carry Bit is not changed unless no device responds.
Accumulator ← Device Address.

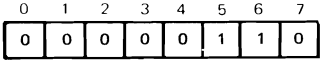
4.4.6 Sense Status SS



An eight-bit status byte from the currently addressed device controller is placed in the Accumulator.

The Carry Bit is not changed unless the device does not respond.
Accumulator ← Status Byte.

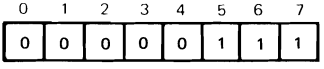
4.4.7 Read Data RD



An eight-bit Data Byte from the currently addressed device controller is placed in the Accumulator.

The Carry Bit is not changed unless the device does not respond.
Accumulator ← Data Byte.

4.4.8 Read Data and Skip RDS



The Status of the currently addressed device controller is sensed and tested.

If the transfer can be made:

When the device is ready, an eight-bit data byte is transferred from the device to the Accumulator. The next program instruction is skipped on the successful execution of this instruction.

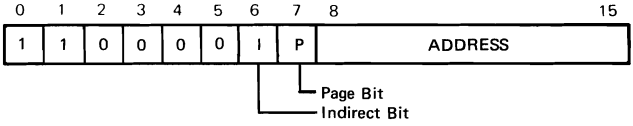
The Carry Bit is not changed unless the device does not respond.
Accumulator ← Data Byte.

If the transfer cannot be made:

If any of bits 5, 6, or 7, of the device status is true and the instruction is aborted, the offending status byte is loaded to the Accumulator, and the next program instruction is not skipped.

The Carry Bit is not changed unless the device does not respond.
Accumulator ◄— Status Byte.

4.4.9 Write Block WB



A block of data bytes is transferred to the addressed device controller from memory beginning at the first effective memory address and continuing sequentially to the last data byte in that Page. The instruction then terminates. On the successful transfer of the complete block of data, the next program instruction is skipped. See Figure 4-6.

The Carry Bit is not changed unless the device does not respond.
Accumulator ◄— undefined.
Addressed Device ◄— Memory (Data Block)

If at any time during the block transfer the device responds with an unusual status, the instruction is aborted, the offending status byte is loaded to the Accumulator, and the next instruction is not skipped.

The Carry Bit is not changed. Accumulator ◄— Status Byte.

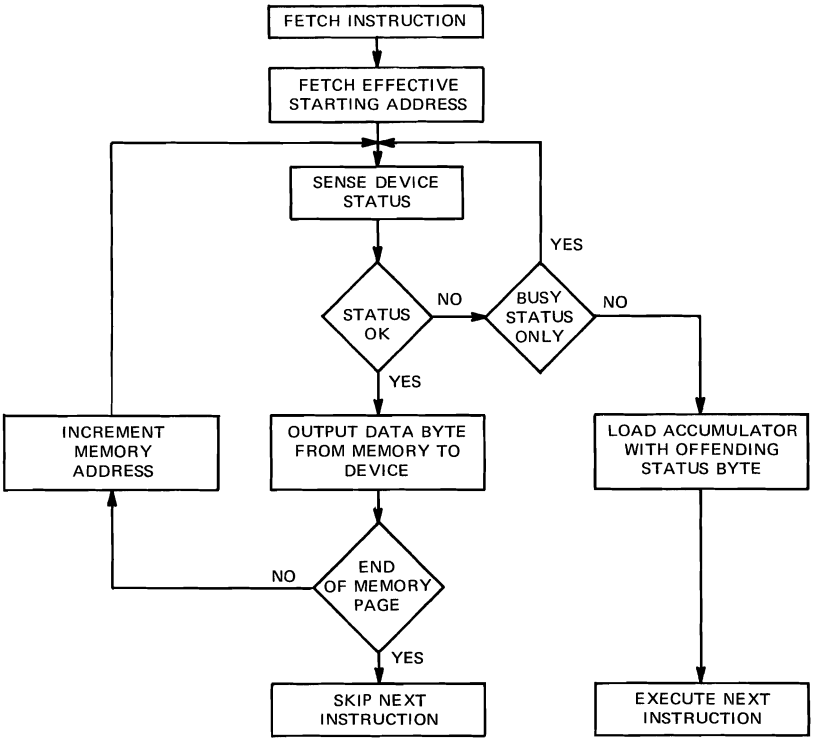
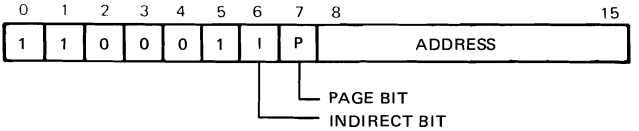


Figure 4-6. Write Block Flow Chart

4.4.10 Read Block RB



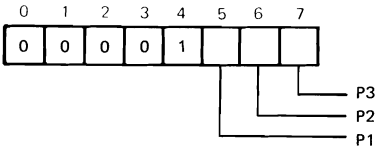
A block of data bytes is transferred from the addressed device controller to memory beginning at the first effective memory address and continuing sequentially to the last memory address in that Page. The instruction then terminates. On the successful transfer of the complete block of data, the next program instruction is skipped.

The Carry Bit is not changed unless the device does not respond.
Accumulator ← undefined.
Memory ← Addressed Device (Data Block)

If at any time during the block transfer the device responds with an unusual status, the instruction is aborted, the offending status byte is loaded to the Accumulator and the next instruction is not skipped.

The Carry Bit is not changed. Accumulator ← Status Byte.

4.4.11 Pulsed I/O PIO P1, P2, P3



This instruction outputs a control level designated PIO, a fixed control pulse designated P0, and any combination of three pulses designated P1, P2, and P3 as specified by the instruction word. The Accumulator is output to the custom interface except for the interval covered by P0. If P3 is specified the Accumulator is loaded from the custom interface on the trailing edge of P3. PIO and P0 are always present during the pulsed I/O instruction.

The Carry Bit is not changed.

Device ← Accumulator

Accumulator ← Data Byte (if P3 is specified)
← Unchanged (if P3 is not specified)

This instruction is intended for custom interface designs. It will not operate with any standard INTERDATA Controllers. Reference is made to Chapter 5 for a detailed explanation of application for this instruction.

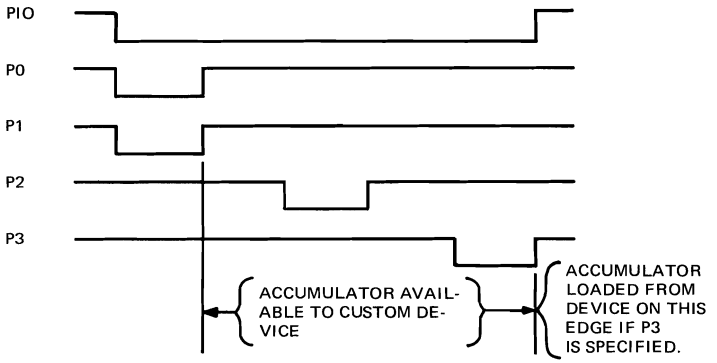


Figure 4-7. Pulsed I/O Timing

4.5 COMMAND INSTRUCTION

The Command Instruction is a Long Form sixteen-bit instruction which is used to set up and control a number of internal machine modes or functions. Two valid modifiers may specify skip conditions on Command Instructions;

NC - Skip if carry not set
NM - Skip if accumulator not minus

The skip conditions are tested after the instruction is executed. If any specified skip condition is true the next program instruction is skipped.

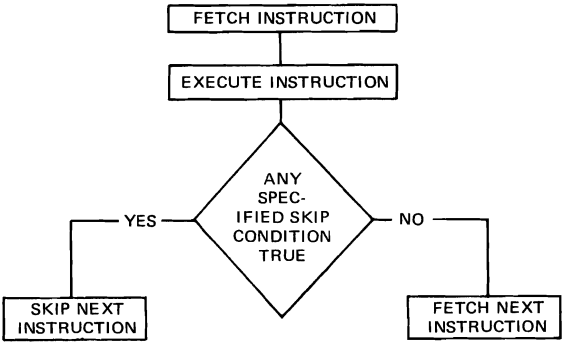
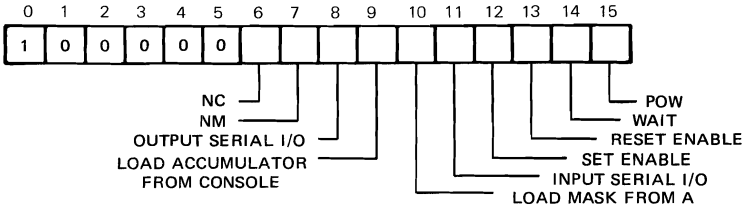


Figure 4-8. Example of Command Instruction

The Command Instruction is not interruptable. It follows that at least one instruction following a Command instruction is always executed before interrupts are recognized regardless of the Master Interrupt Enable Bit or the Individual Interrupt Mask Bits.

4.5.1 Command C



The command functions specified are executed, the skip conditions are tested, and the next program instruction is skipped if any specified skip condition is true.

The Carry Bit may or may not be changed depending on the specified command function(s).

The Accumulator may or may not be changed depending on the specified command functions(s).

BIT 08: Output Serial I/O

The Carry Bit is output to the Serial I/O Port output buffer. Carry is not changed.

BIT 09: Load A

The Accumulator is loaded from the 8 Data Switches (08:15) on the Standard Control Panel.

BIT 10: Load Mask

The Interrupt Mask Register is loaded from the Accumulator. The Accumulator is not changed.

BIT 11: Input Serial I/O

The Carry Bit is loaded from the Serial I/O Port.

BIT 12: Set Enable

The Master Interrupt Enable Bit is set.

BIT 13: Reset Enable

The Master Interrupt Enable Bit is reset.

BIT 14: Wait

The Processor is placed in a WAIT state. Program execution is halted. The Processor remains in the WAIT state until an interrupt is honored or until the Processor is restarted from the Control Panel. In either case the WAIT flip-flop is reset. If an interrupt occurs with the Processor in the WAIT state, the instruction following the Command Instruction is executed before the interrupt operation takes place.

BIT 15: POW

The POW Command causes the Processor to initialize itself in an orderly fashion. This command is issued by the program in the event of a POWER FAIL Interrupt. (The POWER FAIL Interrupt occurs as a result of:

a. Depressing the CLEAR Switch in the Control Panel

b. Switch power OFF from the Control Panel

c. A Primary Power Fail condition if the Power Fail Safe option exists.

The instruction at Location 0000 is executed when power is restored.

NOTE

The following Command Functions are mutually exclusive and may not be combined in any single Command Instruction:

a) SET ENABLE AND RESET ENABLE

b) LOAD MASK AND LOAD A

c) INPUT SERIAL I/O AND OUTPUT SERIAL I/O.

4.6 IMMEDIATE INSTRUCTIONS

Immediate Instructions are Long Form sixteen-bit instructions where the second operand is part of the primary instruction word. The first operand on Logical or Arithmetic operations is contained in the Accumulator. The result of the Immediate operation is ordinarily placed in the Accumulator. Three valid modifiers may specify skip conditions on Immediate operations:

NM - Skip if the result is not minus
NZ - Skip if the result is not zero (valid only on Logical Operations)
NC - Skip if the Carry result is not set (valid only in Arithmetic).

The skip conditions are tested after the Immediate Operation is performed. If any specified skip condition is true the next program instruction is skipped.

A fourth valid modifier is recognized in Immediate Instructions which prevents the result from being loaded into the Accumulator or the Carry Bit. The operation is otherwise carried out so that the skip conditions may be tested. See Figure 4.9.

N - Do not save the result. (The Accumulator and the Carry Bit are not modified. The instruction is otherwise executed and the results tested for specified skip conditions).

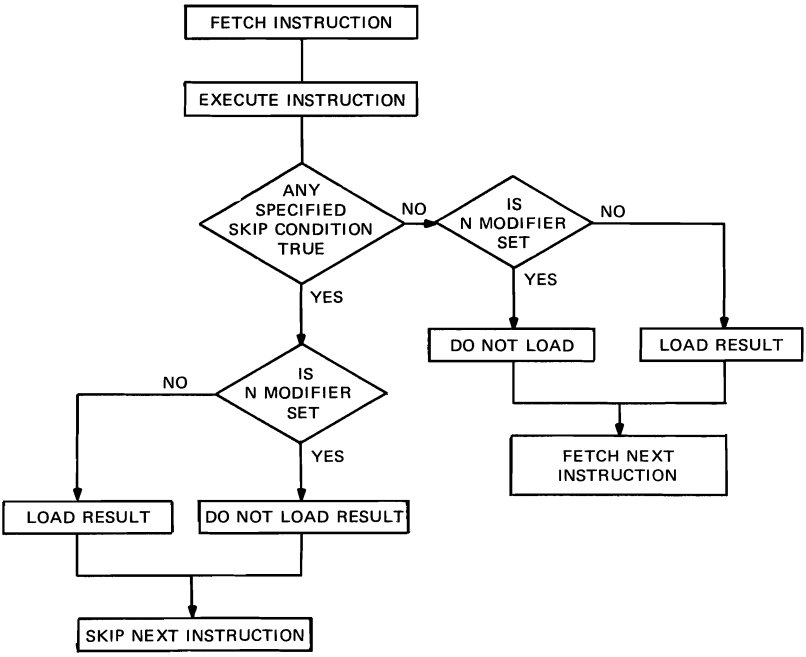
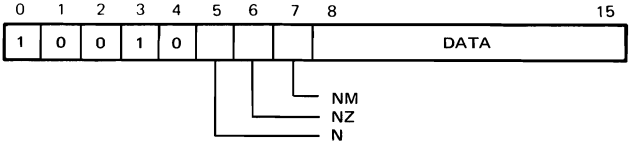


Figure 4-9. Immediate Instructions Flow Chart

4.6.1 Load Immediate LI

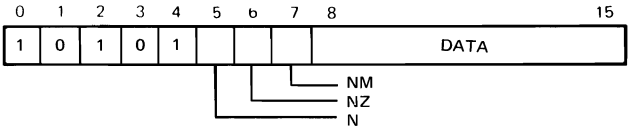


The data byte in the eight-bit DATA field is placed in the Accumulator provided N is reset. If any specified skip conditions are true, the next program instruction is skipped.

The Carry Bit is not changed.

Accumulator ← DATA field (N reset)

4.6.2 AND Immediate NI

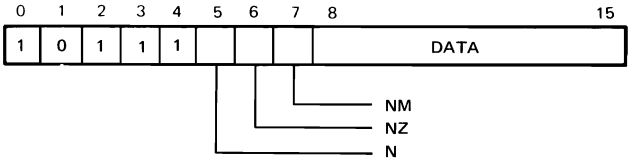


The Accumulator is logically ANDed with the DATA field, the result is placed in the Accumulator provided N is reset. If any specified skip condition is true the next program instruction is skipped.

The Carry Bit is not changed.

Accumulator ← Accumulator ANDed with DATA field (N Reset)

4.6.3 OR Immediate OI

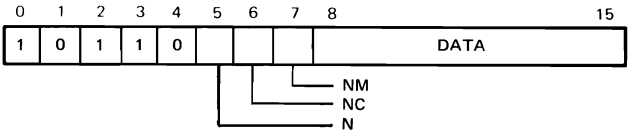


The Accumulator is logically ORed with the eight-bit DATA field, the result is placed in the Accumulator provided N is reset. If any specified skip condition is true the next program instruction is skipped.

The Carry Bit is not changed.

Accumulator ← Accumulator ORed with DATA field (N reset)

4.6.4 Exclusive-OR Immediate XI

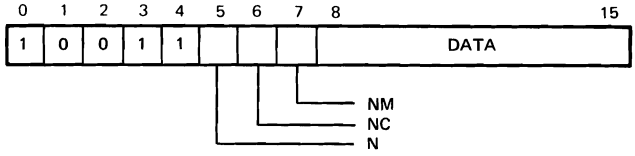


The Accumulator is Exclusive ORed with the eight-bit DATA field, the result is placed in the Accumulator provided N is reset. If any specified skip condition is true the next program instruction is skipped.

The Carry Bit is not changed.

Accumulator ← Accumulator Exclusive ORed with DATA field (N reset)

4.6.5 Add Immediate AI



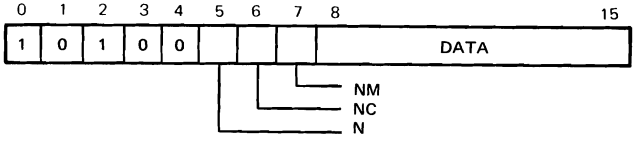
The eight-bit DATA field is added to the Accumulator, the result is placed in the Accumulator provided N is reset. If any specified skip condition is true the next program instruction is skipped.

The Carry Bit ← The Carry Out of the most significant bit of the Accumulator as a result of the addition (N is reset).

Accumulator ← Accumulator plus DATA field (N Reset).

The Carry Bit and the Accumulator are unchanged if N is set.

4.6.6 Subtract Immediate SI



The eight bit DATA field is subtracted from the Accumulator, the result is placed in the Accumulator provided N is reset. If any specified skip condition is true the next program instruction is skipped.

The Carry Bit ← Borrow

The Accumulator ← Accumulator minus the DATA field (N Reset)

The Carry Bit and the Accumulator are unchanged if N is set.

4.7 MEMORY REFERENCE INSTRUCTIONS

Memory Reference Instructions operate in a Page Memory environment. Four separate Addressing Modes are defined: Direct in Page Zero, Direct in the Current Page, Indirect through Page Zero, and Indirect through the Current Page. The Addressing Modes are discussed in detail in Chapter 3. Similarly, a very powerful Auto-Index feature associated with most Memory Reference Instructions is discussed in Chapter 3.

NOTE

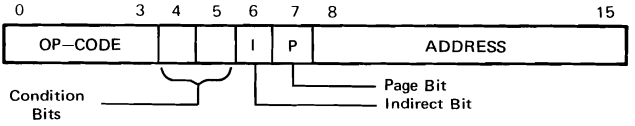
If a Memory Reference Instruction resides on a page boundary then the current page corresponds to the page number of the second byte (the address field) of the instruction word.

Memory Reference Instructions fall into four categories: Branches, Arithmetic and Logical, Bit Operations, and Block I/O Instructions. Any Memory Reference Instruction may operate in any Addressing Mode.

4.7.1 Branches

Branch Instructions are used to direct the program to a new location when certain conditions are met. Branch and Link Instructions are used to direct the program to a new location also. In addition, however, the Branch and Link Instruction stores the Old PSW (Location Count plus the Carry Bit and the Interrupt Enable Bit) in memory for future use by the program.

Branch Instructions take on the general form:

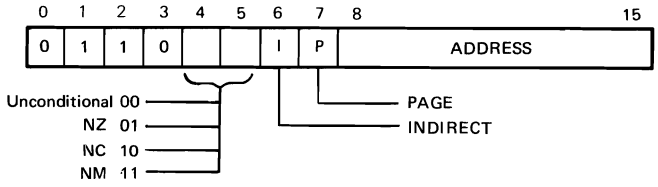


where Bits 4 and 5 are encoded to specify a unique Branch condition.

Bit 4	Bit 5	Valid Modifier	Description
0	0	-	Unconditional.
0	1	NZ	Not Zero - The Branch is taken if the Accumulator is not ZERO.
1	0	NC	Not Carry - The Branch is taken if the Carry Bit is not set.
1	1	NM	Not Minus - The Branch is taken if the Accumulator is not Minus.

If the specified Branch condition is not met, the operation is aborted and the next program instruction is executed.

4.7.1.1 Branch B



The specified Branch condition is tested. If the condition is not true, the operation is aborted and the next program instruction is executed. If the condition is true, the Branch is taken depending on the Addressing Mode as follows:

4.7.1.2 Direct in Page Zero (Bits 6 and 7 = 00)

The Location Counter's byte address is replaced by the eight-bit Address Field of the instruction. The Location Counter's Page Address is set to zero. The Carry Bit, the Enable Bit, and the Accumulator are not changed.

4.7.1.3 Direct in Current Page (Bits 6 and 7 = 01)

The Location Counter's byte address is replaced by the eight-bit Address Field of the instruction. The Location Counter's Page Address is unchanged.

The Carry Bit is not changed. The Accumulator is not changed.

4.7.1.4 Indirect Through Page Zero (Bits 6 and 7 = 10)

The Location Counter is replaced by the two-byte address word in Page Zero. Bit 0 of the Address Word replaces the Carry Bit and Bit 1 of the address word replaces the Interrupt Enable Bit.

Carry ← Bit 0 of the address word.

Accumulator ← Unchanged

Enable ← Bit 1 of the address word.

4.7.1.5 Indirect Through the Current Page (Bits 6 and 7 = 11)

The Location Counter is replaced by the two-byte address word in the Current Page. The Carry Bit is replaced by Bit 0 of the address word. The Interrupt Enable Bit is replaced by Bit 1 of the address word.

Carry ◄——— Bit 0 of the address word.

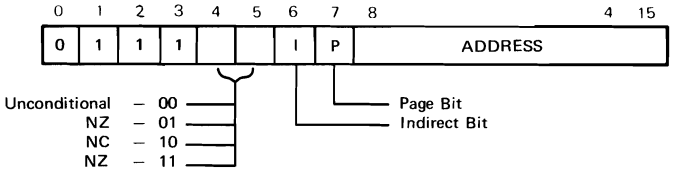
Accumulator ◄—— Unchanged.

Enable ◄——— Bit 1 of the address word.

NOTE

Auto Indexing is not available on Branch Instructions.

4.7.2 Branch and Link BAL



The specified Branch condition is tested. If the condition is not true, the operation is aborted and the next instruction is executed. If the Branch condition is true, the Branch and Link operation is performed depending on the Addressing Mode as described in the following paragraphs.

4.7.2.1 Direct in Page Zero (Bits 6 and 7 = 00)

The Location Counter (corresponding to the next program instruction), the Carry Bit, and the Interrupt Enable Bit are stored in the two-byte memory word in Page Zero indicated by the Address field of the instruction. The Location Counter's byte address is replaced by the Address field plus two. The Location Counter's Page Address is reset to zero.

The Carry Bit, Enable Bit and the Accumulator are not changed.

4.7.2.2 Direct In Current Page

The Location Counter (corresponding to the next program instruction), the Carry Bit, and the Interrupt Enable Bit, are stored in the two-byte memory word in the Current Page indicated by the eight-bit Address field. The Location Counter's byte address is replaced by Address plus two. The Location Counter's Page Address is not changed.

The Carry Bit, Enable Bit and the Accumulator are not changed.

4.7.2.3 Indirect Through Page Zero (Bits 6 and 7 = 10)

The Location Counter (corresponding to the next program instruction), the Carry Bit, and the Interrupt Enable Bit, are stored in the two-byte memory word specified by the address word. (This effective address word itself is in Page Zero at Address and Address plus 1.) The Processor starts executing the instruction at the effective Address plus 2.

The Carry Bit, Enable Bit and the Accumulator are not changed.

4.7.2.4 Indirect Through the Current Page (Bits 6 and 7 = 11)

The Location Counter (corresponding to the next program instruction), the Carry Bit, and the Interrupt Enable Bit are stored in the two-byte memory word specified by the address word. (This effective address word is in the Current Page at Address and Address plus 1.) The Processor starts executing the instruction at the effective Address plus 2.

The Carry Bit, Enable Bit and the Accumulator are not changed.

NOTE

Auto Indexing is not available on Branch and Link Instructions.

4.7.3 Arithmetic and Logical Memory Reference Instructions

Arithmetic and Logical Memory Reference Instructions operate on the Accumulator with a data byte from memory. Any of four Addressing Modes; Direct in Page Zero, Direct in Current Page, Indirect through Page Zero, or Indirect through the Current Page, may be specified by the instruction word. Auto-Indexing may be specified on all Indirect Memory Referenced Instructions except Branches. (See Chapter 3 for details on Addressing Modes and Auto-Indexing.)

One valid modifier is associated with Arithmetic and Logical Memory Reference Instructions. The modifier takes on a different meaning depending on the operation code as follows: If the operation is ADD or SUBTRACT,

N = Skip on No Carry and don't load result into Accumulator

The operation is performed and the Carry Result is tested. If the Carry result is a ONE, the next instruction is executed. If the Carry Result is a ZERO, the next instruction is skipped. In any case, the Carry Bit and the Accumulator are not changed.

If the operation is AND, OR, Exclusive OR

N = Skip if the Logical Result is not ZERO and don't load result into Accumulator.

The operation is performed and the logical result is tested. If the result is ZERO, the next instruction is fetched. If the result is not ZERO, the next instruction is skipped. In either case the Carry Bit and the Accumulator are not changed. See Figure 4-10.

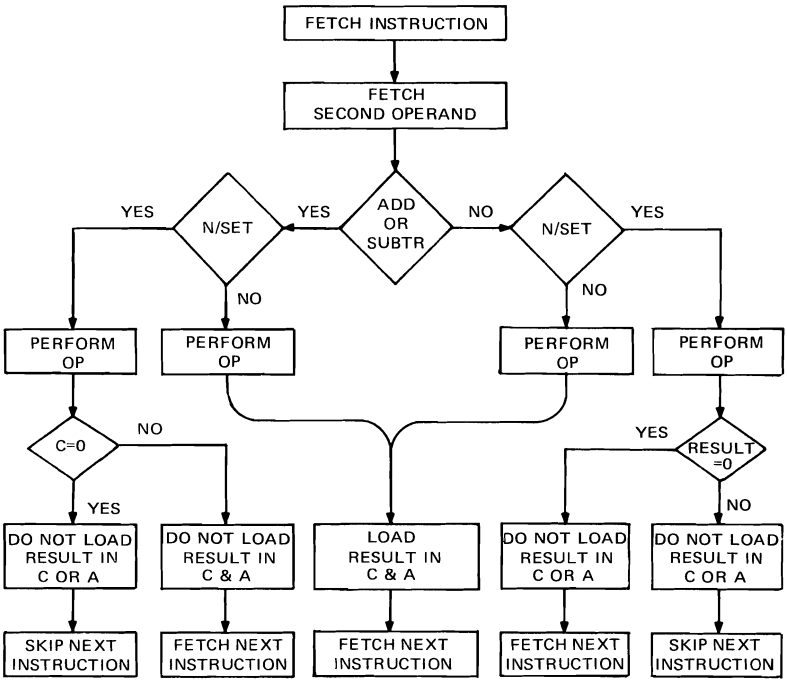
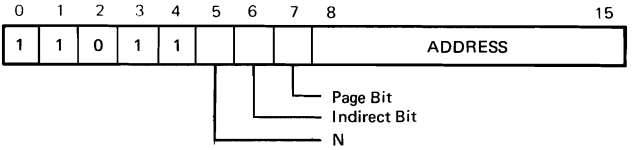


Figure 4-10. Arithmetic and Logical Memory Reference Instructions Flow Chart

4.7.3.1 Add A



If N = 0, the second operand is Added to the Accumulator, the result is placed in the Accumulator, and the next instruction is fetched.

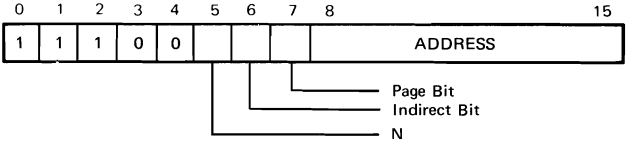
Carry Bit ← The Carry Out resulting from the ADD Operation.

Accumulator ← Accumulator plus the second operand.

If N = 1, the Carry result of the Add operation is tested. If the Carry result is ZERO, the next program instruction is skipped. If the Carry result is ONE, the next program instruction is not skipped. In either case, the Carry Bit and the Accumulator are not changed.

Carry Bit is not changed. Accumulator is not changed.

4.7.3.2 Subtract S



If N = 0, the second operand is subtracted from the Accumulator. The result is placed in the Accumulator, and the next instruction is fetched.

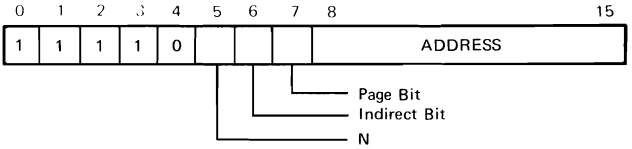
Carry Bit ← The Carry (Borrow) resulting from the Subtract operation.

Accumulator ← Accumulator minus the second operand.

If N = 1, the Carry result of the Subtract operation is tested. If the Carry result is ZERO, the next program instruction is skipped. If the Carry result is ONE, the next program instruction is not skipped. In either case, the Carry Bit and the Accumulator are not changed.

Carry Bit is not changed. Accumulator is not changed.

4.7.3.3 Exclusive--OR X



If N = 0, the second operand is Exclusive ORed with the Accumulator, the result of the operation is placed in the Accumulator.

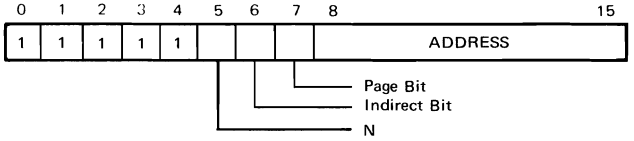
Carry Bit is not changed.

Accumulator ← Accumulator Exclusive ORed with the Second Operand

If N = 1, the logical result of the operation is tested. If the result is ZERO, the next instruction is not skipped. If the result is not ZERO, the next instruction is skipped. In either case the Accumulator is not changed.

Carry Bit is not changed. Accumulator is not changed.

4.7.3.4 OR O



If N = 0, the second operand is logically ORed with the Accumulator, the result is placed in the Accumulator.

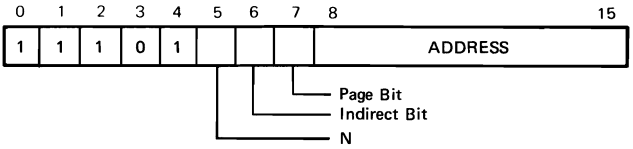
The Carry Bit is not changed.

Accumulator ←Accumulator ORed with the Second Operand

If N = 1, the logical result of the operation is tested. If the result is not ZERO, the next instruction is skipped. If the result is ZERO, the next instruction is not skipped. In either case the Accumulator is not changed.

Carry Bit is not changed. Accumulator is not changed.

4.7.3.5 AND N



If N = 0, the second operand is logically ANDed with the Accumulator and the result is placed in the Accumulator.

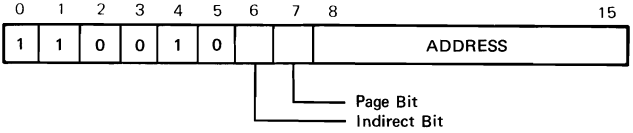
The Carry Bit is not changed.

Accumulator ← Accumulator ANDed with the Second Operand

If N = 1, the logical result of the operation is tested. If the result is not ZERO, the next instruction is skipped. If the result is ZERO, the next instruction is not skipped. In either case the Accumulator is not changed.

Carry Bit is not changed. Accumulator is not changed.

4.7.3.6 Store ST

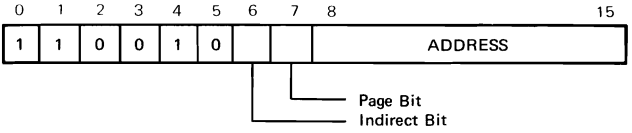


The Accumulator is stored at the effective memory address.

The Carry Bit is not changed. The Accumulator is not changed.

Effective Address ← Accumulator

4.7.3.7 Load L

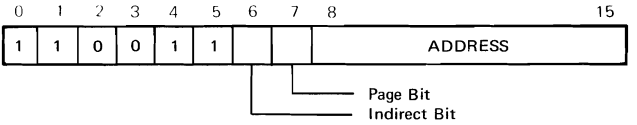


The Accumulator is loaded from the effective address.

The Carry Bit is not changed.

Accumulator ← Data Byte from effective address.

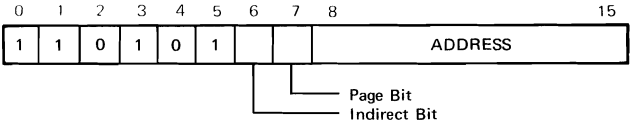
4.7.3.8 Increment and Skip on Not Zero ISN



The content of the effective address is incremented. If the incremented value does not equal X'00', the next two-byte instruction is skipped.

The Carry Bit is not changed. The Accumulator is not changed.

4.7.3.9 Increment and Skip on Zero ISZ



The contents of the effective address is incremented. If the incremented value equals X'00', the next two-byte instruction is skipped.

The Carry Bit is not changed. The Accumulator is not changed.

4.7.4 Bit Operation Memory Reference Instruction

The Bit Operation instructions are effective for evaluating Boolean expressions where the state of the variables in the expression is represented by individual bits in memory.

Bit Operation instructions involve two single bit operands which may be ORed or ANDed with the result placed in the Carry Bit. The first bit operand is the Carry Bit:

First Bit Operand = Carry Bit

The second bit operand is derived from the Accumulator and the addressed memory location as follows:

$$\text{Second Bit Operand} = A_0 \cdot M_0 + A_1 \cdot M_1 + \ldots + A_7 \cdot M_7$$

where A_n is bit n of the Accumulator and
where M_n is bit n of the Addressed Memory Location

Example: If

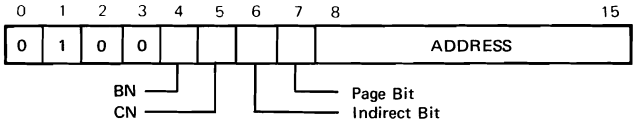
Accumulator	=	10100001
Memory Data	=	10010111
then		
Second Bit Operand	=	$1 \cdot 1+0 \cdot 0+1 \cdot 0+0 \cdot 1+0 \cdot 0+0 \cdot 1+0 \cdot 1+1 \cdot 1$
	=	1+0+0+0+0+0+0+1
	=	1

Two valid modifiers are associated with the Bit Operation instructions:

BN - Bit Not. The second bit operand is complemented before it is used.

CN - Carry Not. The Carry Bit (first bit operand) is complemented before it is used.

4.7.4.1 AND BIT NB



The Second operand (or its complement if BN=1) is ANDed with the Carry Bit (or its complement if CN=1). The result is loaded into the Carry Bit.

Accumulator ⟵ Unchanged

Carry Bit ⟵ Carry Bit • Second Operand subject to modifiers

Example:

Suppose the Accumulator = 11001011
and the Carry Bit = 1
and we execute
NB BYTE
where the contents of BYTE = 10110001

Then, the first operand (Carry Bit) = 1

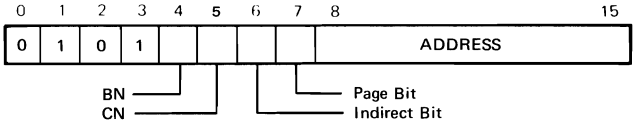
The second operand, derived by computing
$A_0 \cdot M_0 + A_1 \cdot M_1 + \ldots A_7 M_7$ = 1

And the result loaded to the Carry Bit = 0
The Accumulator is not changed.

Example:

Suppose the Accumulator	=	11001011
and the Carry Bit	=	1
and we execute NB CN, BYTE		
where the contents of BYTE	=	10110001
Then, the first operand (Carry Bit inverted)	=	0
The second operand, (as above)	=	1
And the result loaded to the Carry Bit	=	0
The Accumulator is not changed.		

4.7.4.2 OR BIT OB



The second bit operand (or its complement if BN=1) is ORed with the Carry Bit (or its complement if CN=1). The result is loaded in the Carry Bit.

Accumulator ←—Unchanged

Carry Bit ←— Carry + Second Operand subject to modifiers

Example:

Suppose the Accumulator	=	10111010
and the Carry Bit	=	0
and we execute OB BN, CN BYTE		
where the contents of BYTE	=	01000101
Then, the first operand (Carry Bit inverted)	=	1
and the second bit operand is		
$\overline{(A_0M_0+A_1M_1+\ldots A_7M_7)}$	=	1
and the result loaded to the Carry Bit	=	1
The Accumulator is not changed.		

CHAPTER 5

INPUT/OUTPUT (I/O) SYSTEMS

5.1 INTRODUCTION

This chapter describes the Model 1 I/O System. There are several methods of communication between the INTERDATA Processor and peripheral devices. The methods vary in speed, sophistication, and the amount of attention required by the Processor. Thus, the Systems Interface may be tailored to communicate efficiently with all types of peripheral devices.

There are two primary purposes for this chapter: to familiarize the reader with the INTERDATA Systems Interface, and to provide the data required to effectively interface peripheral equipment to INTERDATA Model 1 Digital Systems. A functional description of each I/O sub-system is provided later in this chapter, followed by a physical description of the layout and interconnection of a typical system. The coding and sequence of operation of all I/O Instructions, the considerations and specifications in designing device controllers, and a General Purpose Interface Controller available from INTERDATA to facilitate custom interface design are described.

5.2 I/O SYSTEM BLOCK DIAGRAM ANALYSIS

Figure 5-1 is a block diagram of an INTERDATA Digital System emphasizing the Systems Interface capability. Note that there are four separate methods of communicating with peripheral devices or systems:

1. The Multiplexor Channel
2. A Selector Channel
3. Universal Memory Bus Interface (UMBI)
4. Direct Memory Access

Each of the four methods communicates via a bus with device controllers. The Systems Interface can communicate with up to 256 devices. The following paragraphs describe each of the interface methods.

5.2.1 Multiplexor Channel

Figure 5-2 is a block diagram of the Multiplexor Channel. The Multiplexor Channel is a byte oriented I/O system which communicates directly with up to 256 peripheral devices. The Multiplexor Bus consists of 27 lines; eight data input, eight data output, eight control lines, two test lines, and an Initialize line. The two test input lines from the device controllers are Sync (SYN) and Attention (ATN). The final line is System Clear (SCLR) to all device controllers.

The lines in the Multiplexor Bus are listed below:

Data Available Lines	DAL00:07	(Processor → Device)	8 Lines
Data Request Lines	DRL00:08	(Processor ← Device)	8 Lines
Control Lines	SR	(Processor → Device)	1
	DR	(Processor → Device)	1
	CMD	(Processor → Device)	1
	DA	(Processor → Device)	1
	ADRS	(Processor → Device)	1
	ACK	(Processor → Device)	1
	CL06	(Processor → Device)	1
	CL07	(Processor → Device)	1
Test Lines	ATN	(Processor ← Device)	1
	SYN	(Processor ← Device)	1
Initalize	SCLR0	(Processor → Device)	1

The following general definitions apply:

Data Available Lines DAL00:07: The Data Available Lines are used to transfer one byte of Address, Command, or Data from the Processor for the Device when accompanied by the appropriate control line (ADRS, CMD, or DA respectively).

Data Request Lines DRL00:07: The Data Request Lines are used to transfer one byte of Address, Status, or Data from the Device to the Processor when the Processor issues the appropriate control line (ACK, SR, or DR respectively).

Control Lines

SR: Status Request.	The device controller must present device status to the DRL lines, followed by a SYN.
DR: Data Request.	The device controller presents data on the DRL lines, followed by a SYN.

ACK: Acknowledge.	Device controller presents it's device address on the DRL line, followed by a SYN.
DA: Data Available.	The Processor presents data on the DAL lines, for transfer to the device, and then issues a DA.
CMD: Command.	The Processor presents a command on the DAL lines, and issues a CMD.
ADRS: Address.	The Processor presents an address to the DAL lines, and issues an ADRS.
ATN: Attention.	Any device desiring to interrupt the Processor will activate the ATN line and will hold this line until an ACK is received from the Processor. Then the device controller presents the device address to the DRL line, followed by a SYN.
SYN:	This signal is generated by the device to inform the Processor that it has properly responded to a control signal. Any Processor control signal, except SCLR, must be followed by a SYN signal before the Processor will proceed. This provides a reliable REQUEST/RESPONSE mode of operation.
SCLR: System Clear.	This is a metalic contact to ground that occurs during power fail, power up, or initializing the machine.

NOTE

All lines, except ACK, are connected in parallel to all devices. The ACK line is connected in series with all devices. The ACK signal, generated by the Processor, is sent to the first device on the bus. If it has an interrupt pending (i.e. it has generated an ATN), it will not pass the ACK signal on to the next device and will respond as in definition of ACK. If there is no interrupt pending, ACK is passed to the next device.

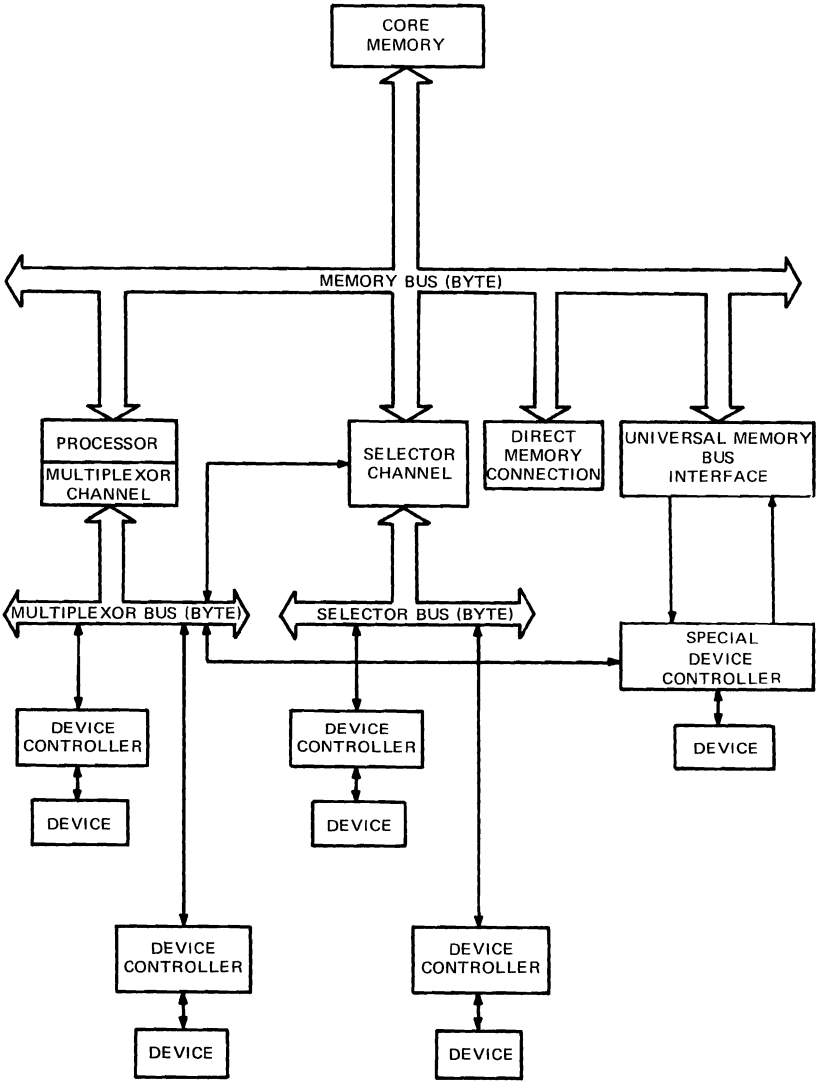


Figure 5-1. Systems Interface, Block Diagram

There are two modes of communicating over the Multiplexor Bus; asynchronous and pulsed. Asynchronous mode is used in virtually all standard INTERDATA device controllers. The pulsed mode is very useful for custom controller designs described later in the chapter.

A typical sequence of operations over the Multiplexor Channel is:

1. The Processor addresses a device controller over the eight System Data Lines. The address appears on the bus to all device controllers, and the Processor activates a Control

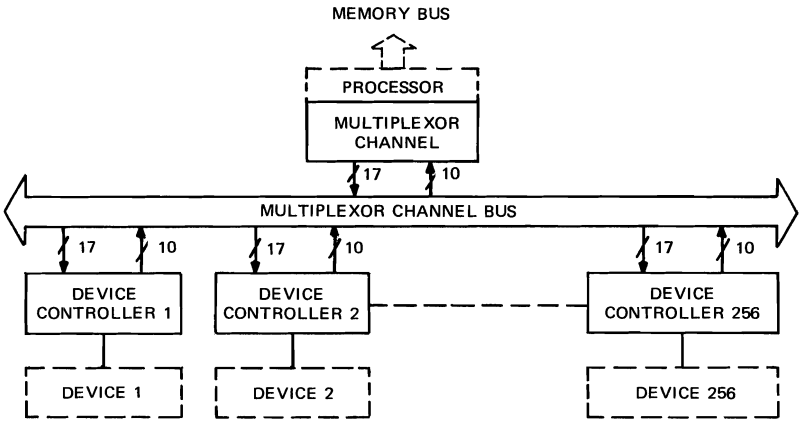


Figure 5-2. Multiplexor Channel, Block Diagram

Line which specifies that the Data Lines provide an address (rather than data).

2. The device controllers use the Control Line to enable address decoders. Each device controller decodes its own address. Assuming that the Data Lines are providing the address of one of the device controllers tied to the Multiplexor Channel, the device controller decodes its address and responds by sending the SYN signal back to the Multiplexor Channel. That device is now in the addressed (on-line) state, and will respond to subsequent activity on the Multiplexor Bus.

3. The Processor may now change the Control and Data Lines. The device controller remains addressed until another device controller is addressed or until a System Clear (SCLR) signal is received.

4. If the next operation is an output operation, the Processor provides the byte of data on the Data Lines, followed by a Control Line indicating an output operation. The device controller responds with the SYN signal when it has accepted the byte.

5. If the next operation is an input operation, the Processor raises a Control Line which requests a byte of data. The device controller sends a byte to the Processor via the Data Lines. The SYN signal is sent to indicate that the data is ready.

The sequence provided here is simplified. The entire sequence for each type of instruction is listed later in the chapter. The final line to be introduced here is the Attention (ATN) line. The Attention line provides a means of interrupting the Processor through one of the priority interrupt lines built into the Processor. Each controller has an interrupt Queue flip-flop which may be set by conditions within either the device or the device controller. The output from the Queue flip-flop is sent to the Multiplexor Channel as ATN. ATN may be initiated by any device controller, at any time, whether it is addressed or not. The program initiates a hardware scan cycle to determine which device controller caused the ATN signal. The highest priority interrupting device automatically returns its device number to the Processor. This interrupt feature is described in more detail later in this chapter.

5.2.2 Selector Channel

The optional Selector Channel provides block data transfer between one of up to 25 I/O devices, and memory. Once initiated, the transfer is independent of the Processor. The Processor specifies the device address, the type of operation (Read or Write), the starting address in memory, and the number of bytes to transfer. The Selector Channel then completes the transfer, cycle stealing from the Processor, without further direction by the Processor. Upon completion of the transfer, or termination of the transfer due to a fault, the Selector Channel Busy condition is dropped and the Processor is notified via an interrupt.

Figure 5-3 is a Block Diagram of the Selector Channel. Address lines to, and data lines to and from, the Memory Bus are shown on the right side. The Memory Bus Control Logic (one of several arbitrary functional groupings used only for purposes of this block diagram) gates an address to the Memory Bus and data to or from the bus depending upon the direction of transfer. The Selector Channel Data Register (DR) stores the eight-bit data byte to/from memory. The transfer Control Logic gates the data between the Selector Bus (shown on the bottom) and the Data Register in eight-bit bytes. The address circuits are shown in the upper right area. The Byte Count Register (BC) is loaded in two bytes from the Multiplexor Bus. The Address Register (AR) is loaded with the starting address in two bytes. After each byte of data transfers to/from memory, the BC is decremented and its contents checked against zero. When BC = 0, a terminate signal is

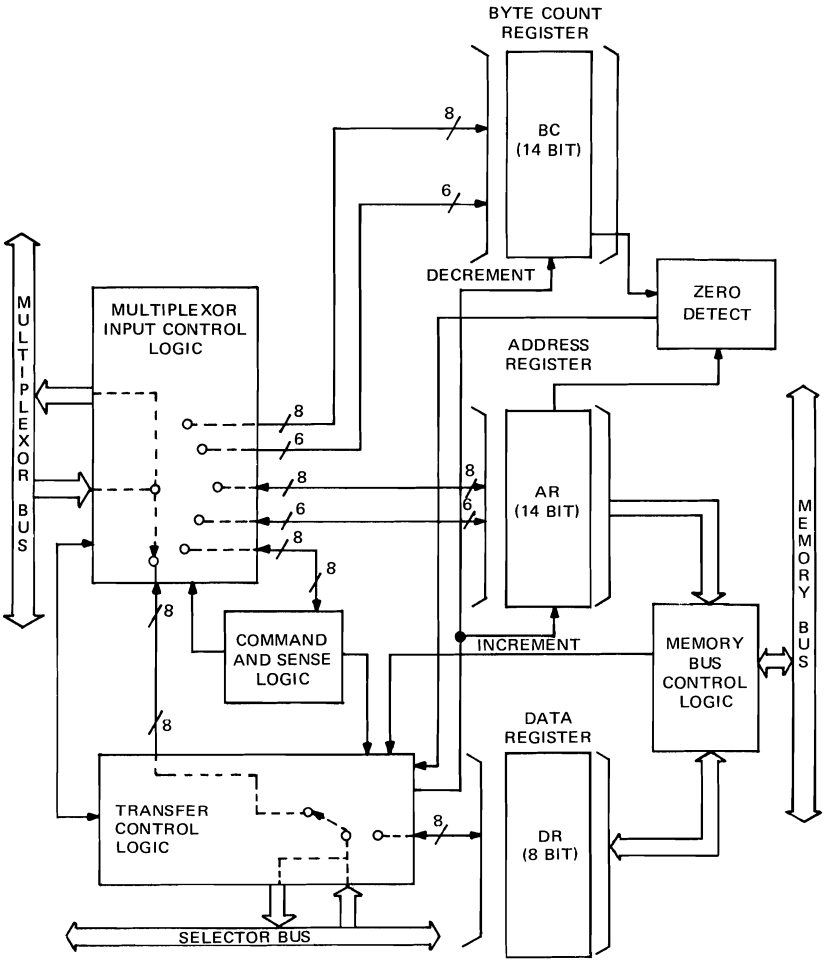


Figure 5-3. Selector Channel, Block Diagram

sent to the Transfer Control Logic. If BC$\neq$0, the next byte transfer is initiated to/from the next sequential memory address.

The Multiplexor Bus is shown on the left side of Figure 5-3. Note that the Multiplexor Bus may be gated to any one of the six places. The gates are functionally represented by a six position rotary switch. With the gating as shown, and assuming the Transfer Control Logic is also as shown, the Multiplexor Bus is gated directly to the Selector Bus. This is the condition which exists when the Selector Channel has

not been addressed. Thus, all devices on the Selector Channel may be used via the Multiplexor Channel if the Selector Channel is not in use. (Of course, the device must be capable of operating within the Multiplexor Channel timing constraints.) Four of the remaining five points onto which the Multiplexor Bus may be gated, are the Upper and Lower halfs of BC and AR. The sixth point is designated Command and Sense Logic. Commands from the Processor are decoded in this block to produce control signals for the Transfer Control Logic and the Multiplexor Input Control Logic.

The following is a typical sequence of operation for a Selector Channel I/O Operation. Figure 5-4 is a flow chart of Selector Channel operation. Circled numbers on Figure 5-4 refer to steps in the following sequence:

1. The device controller is addressed and the appropriate command sent to it (for example, Read Tape Forward).

2. The Selector Channel AR and BC are loaded via four byte transfers from the Multiplexor Channel. The BC may be odd or even.

NOTE

Steps 1 and 2 may be reversed.

3. A command which specifies if this is an input operation (information received from the device) is sent to the Selector Channel from the Multiplexor Channel. (The Selector Channel is initialized and returns to the output state (information to the device) on completion of transfer).

4. A GO Command which starts the transfer operation is sent from the Multiplexor Channel to the Selector Channel.

NOTE

The Processor is now free to continue its program while the block I/O transfer is performed by the Selector Channel on a cycle-stealing basis. Steps 5 through 8 apply solely to Read operations (memory to device). Steps 9 through 12 apply to Write operations (device to memory).

5. If this is a Read operation, the Selector Channel requests memory service via the built-in memory port in the Processor. When service is granted, the Selector Channel fetches a byte from memory.

6. When memory data becomes available, it is gated to the DR.

7. The Status Byte from the device is examined. If the device Busy Bit (Bit 4) is true, the Selector Channel waits for it to become false. If any of the status Bits 5, 6, or 7 are true, the transfer is terminated.

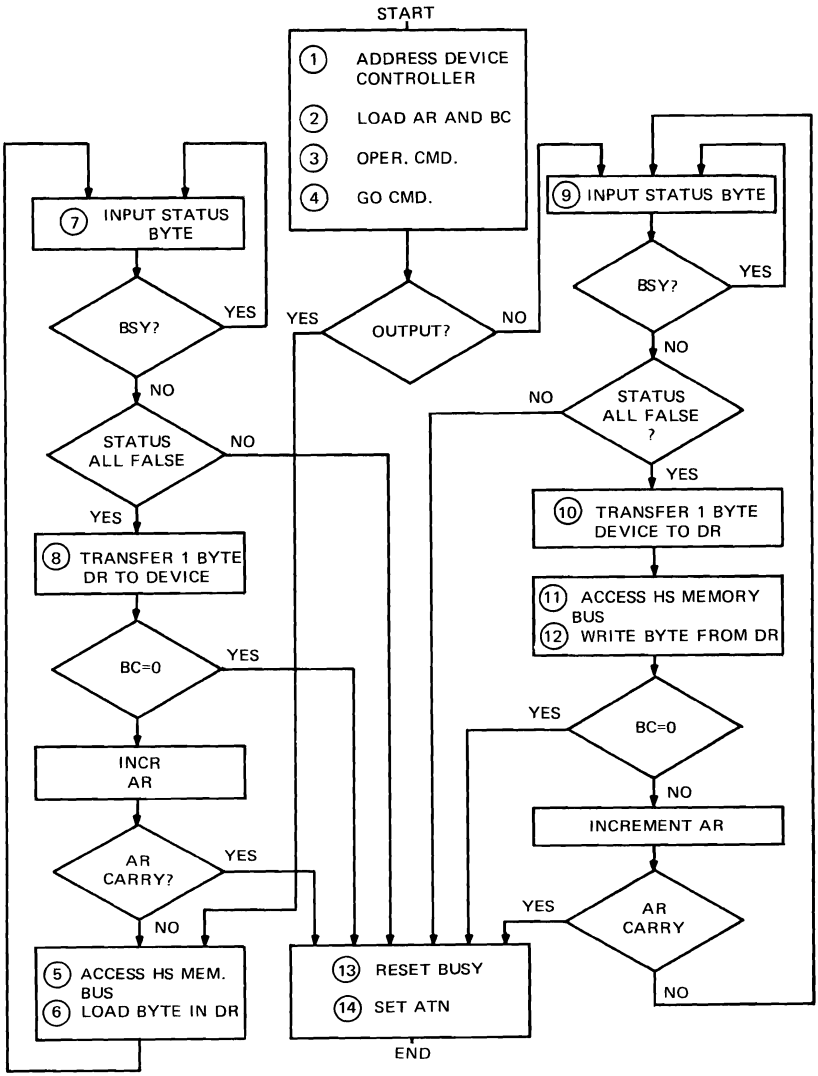


Figure 5-4. Selector Channel, Flow Chart

8. A byte is transferred from the DR to the device when the device is not busy. If the BC is zero, the transfer is terminated. The AR is incremented and if there is a carry (AR=0) the transfer is terminated.

9. If this is a Write Operation, the Status Byte is input from the device. If the Busy is true, the Selector Channel waits for it to become false. If any of the other three bits in the status code are true, the transfer sequence is terminated. If all bits are false, the sequence continues.

10. A byte is transferred from the device to the DR.

11. The Selector Channel requests memory service.

12. The byte in DR is written into the addressed memory location when granted memory.

    The AR is incremented and BC is decremented. If there is a Carry from AR or BC is zero, the transfer is terminated, otherwise the sequence returns to Step 9.

    Steps 13 through 15 describe the termination sequence. Any of the following conditions will cause termination:

    a. BC equals ZERO

    b. AR increments to ZERO (carry out of AR).

    c. A status failure from the device (EX, EOM, or DU).

    d. A Stop Command from the Processor.

NOTE

If the Selector Channel is in a memory cycle when the Stop Command is received from the Processor, execution of Stop Command will be delayed until the completion of the memory cycle.

13. Reset the Selector Channel Busy indication.

14. Set the Selector Channel Attention flip-flop to generate an interrupt to the Processor.

15. After the Processor acknowledges the interrupt and addresses the Selector Channel, it will send a Status Request to the Selector Channel which will check the status code of the Selector Channel. If the transfer terminated with BC not ZERO, Bit 6 will be set. The contents of BC may be read by issuing two Data Requests which return the most, and then the least, significant bytes respectively.

The Selector Channel is complete on one mother-board which is mounted in a Universal Expansion Slot. Every Universal Expansion Slot is wired uniformly to accommodate either a memory module, a standard INTERDATA device controller, or a Selector Channel, or a Universal Memory Bus Interface, or a customer designed Direct Memory Connection. Column 0 of both the top and bottom connector (fields 1 and 0) are reserved for use by the memory system. Columns 1 and 2 of the bottom connector are reserved for the standard Multiplexor Bus wiring. The same two columns on the top connector are wired with a convenient stitch pattern useful for interconnecting two adjacent mother-boards. Ordinarily, any standard INTERDATA device controller may

be plugged into any Universal Expansion Slot without a need to modify the back panel wiring. Core memory modules and ROM modules require no back panel wiring changes.

5.2.3 Universal Memory Bus Interface (UMBI)

The optional Universal Memory Bus Interface (UMBI) permits an eight-bit data transfer between memory and an external device, without transferring the data through the Processor. The UMBI is not a stand-alone circuit, but should be considered part of a special device controller for communicating with the memory. It contains standard gating and control circuits made up of IC logic mounted directly on the motherboard. The remainder of the board is available for the user's designed circuit. It has space for dual in line IC's and discrete axial lead components of the 1/4 watt resistor size. Wire wrap pins provide for interconnections. The back panel connector (Connector 1, top) provides access to the standard expansion slot stitch pattern if required to communicate with other boards.

NOTE

The UMBI has access to both the Multiplexor Bus and the Memory Bus on one card.

As shown in Figure 5-5, the customer designed portion of the controller provides fourteen bits of address, provides eight bits of data for a Memory Write operation, and receives eight bits of data on a Memory Read operation. Control is exercised by means of the Start pulse (START0), the Memory Write Line (W1), and the Initialize signal (CLR1). The UMBI provides the customer circuit with a positive pulse (DR1) which may be used to strobe the DAXX1 lines when a byte is available from memory. The Control Lines, BSY0 and SEL0, indicate when the Data and Address Registers may be modified.

A typical controller would be made up of the UMBI for Memory Bus and Multiplexor Bus access circuits plus any number of wire-wrapped IC boards the particular application requires. Interconnections are made with the Back Panel stitch pattern and with cables if necessary.

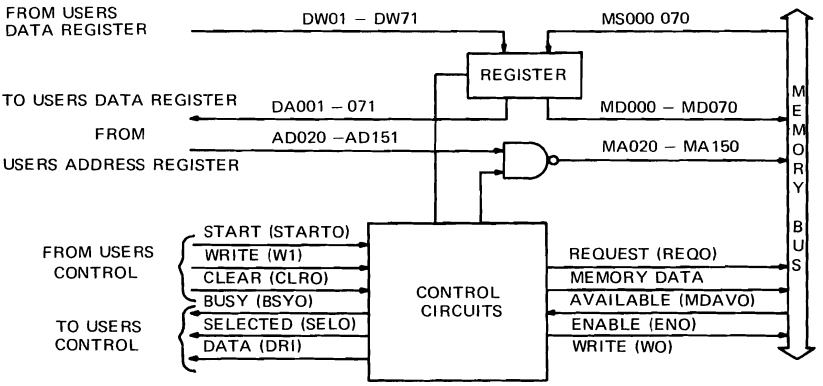
5.2.4 Direct Memory Connection

The user may connect directly to the Memory Bus without the use of the UMBI. The timing and interface rules for the bus must be adhered to. This can be useful if special requirements, such as incrementing data registers, are desired. See Chapter 6 for more detail.

5.3 INPUT/OUTPUT INSTRUCTIONS

5.3.1 Introduction

This section describes the Model 1 Input/Output (I/O) instructions which operate in the Asynchromous Mode. Each instruction is implemented by a sequence of request/response type operations generated automatically by the hardware.



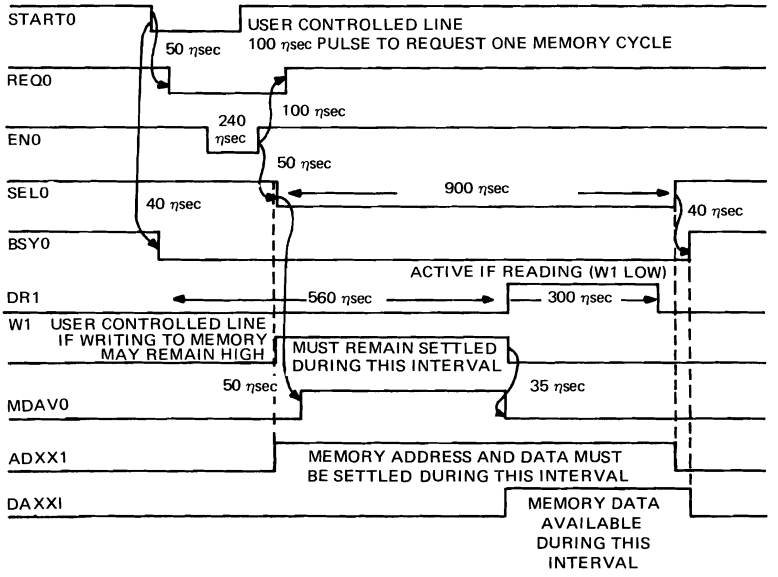
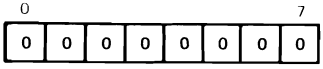


Figure 5-5. Universal Memory Bus Interface, Block Diagram

5.3.2 Address ADRS



The device controller logic for the Address Instruction is shown in Figure 5-6.

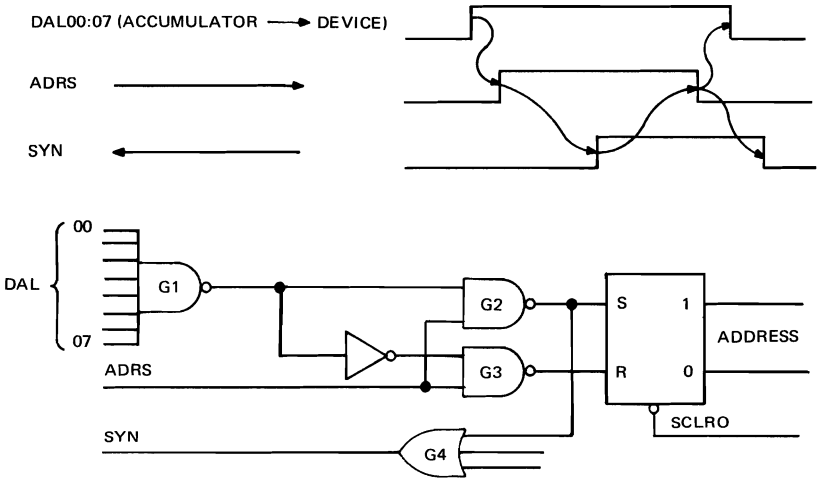


Figure 5-6. Device Controller Logic For The Address Instruction

1. The Processor places the Accumulator, which contains the device number, on the eight Data Available Lines (DAL00:07)
2. The Processor activates the Address (ADRS) control line.
3. The device controller which decodes its address via Gate G1 sets its Address flip-flop through Gate G2 (Since the address decoded at Gate G1 is unique to each device controller the Address flip-flop for all other device controllers is reset by this instruction).
4. The output from Gate G2, via OR Gate G4, also raises the Synchronization (SYN) response from the device controller to the Processor. This indicates that the device controller has recognized the Address Instruction.

5. When the Processor receives the SYN signal, it first removes the Address Control line and then the device number. The device in turn removes the SYN signal, and the instruction is completed.

NOTE

Only one device controller may have its Address Flip-flop set at any time. The device controller will remain addressed until a different address or until a System Clear (SCLRO) signal is generated. Once a device controller is addressed it responds to all Input/Output instructions on the Multiplexor Bus. The user should keep this in mind when responding to interrupts.

NOTE

A time out feature is provided in the Processor to prevent locking up the computer, waiting for SYN, because of a malfunctioning device or a nonexistent device. The time out signal (referred to as False Sync) is generated if the device does not respond with SYN in approximately 30 ± 10 microseconds, if a False SYN occurs, the Processor aborts the instruction and unconditionally sets the Carry Bit.

NOTE

The Processor guarantees the width of the ADRS Control line to be 350 nanoseconds minimum. The designer must design the controller such that when some other device is addressed, the previously addressed device will clear its ADRS flip-flop within the 350 ns period.

5.3.3 Output Command OC

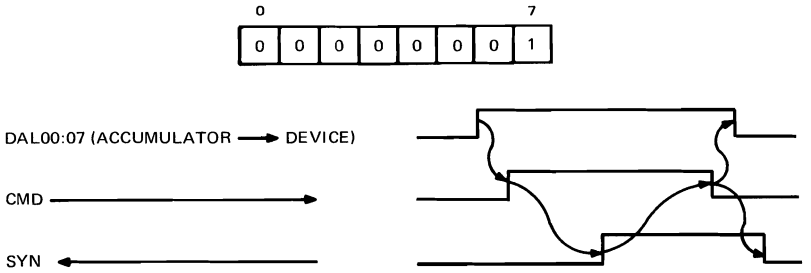


Figure 5-7 provides the device controller logic for the output Command Instruction.

1. The Processor outputs the Accumulator which contains the Command Byte on the eight Data Available lines.

2. The Processor activates the Output Command (CMD) control line.

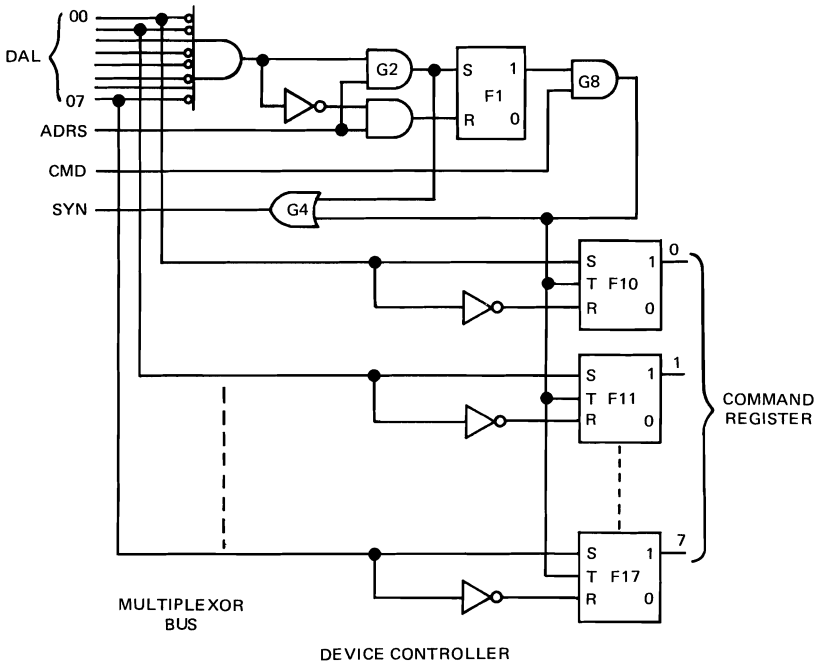


Figure 5-7. Device Controller Logic for the Output Command Instruction

3. The addressed device controller receives the eight bit Command Byte and loads it into the Command Register (F10-F17) using the output from Gate G8. The output from gate G8 also sends a SYN back to the Processor through Gate G4.

4. When the Processor receives the SYN signal it removes the CMD control line and the Command Byte, the device in turn removes the SYN signal and the instruction is complete.

5.3.4 Write Data WD

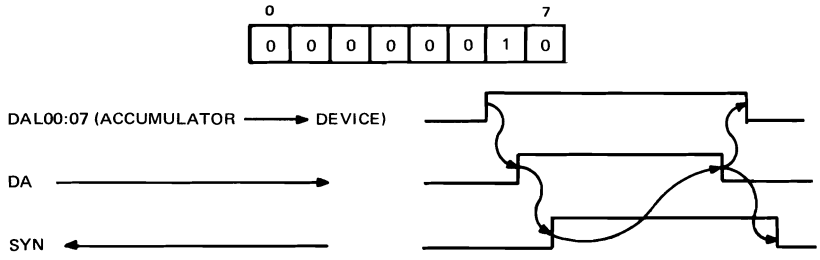


Figure 5-8 provides the device controller logic for the Write Data Instruction.

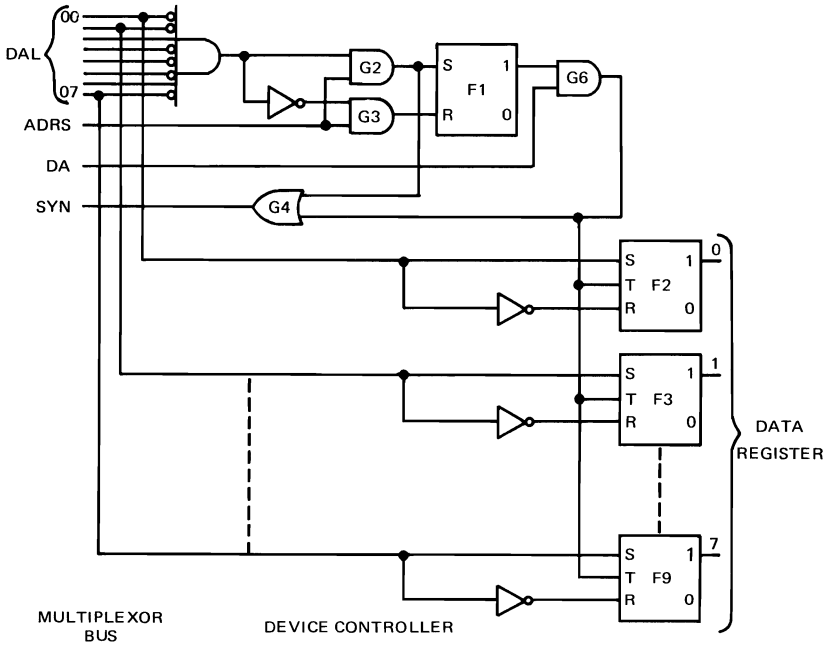
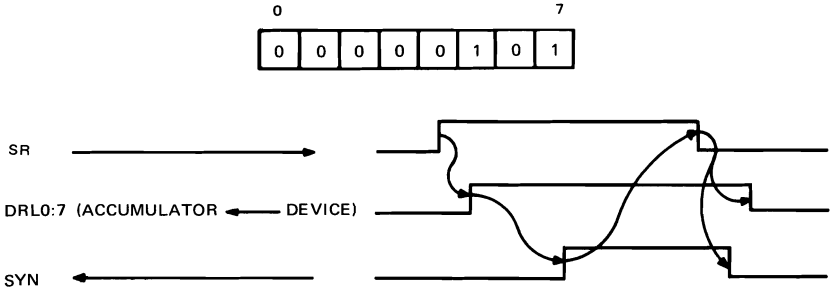


Figure 5-8. Device Controller Logic for the Write Data Instruction

1. The Processor outputs the Accumulator which contains the eight bit Data Byte on the Data Available lines.
2. The Processor activates the Data Available (DA) control line.
3. The currently addressed device controller receives the eight bit Data Byte and loads it into the device controller Data Register (F2 through F9), using the output from Gate G6.
4. The G6 output is also used to return SYN to the Processor through Gate G4.
5. When the Processor receives the SYN signal it removes the DA control line and the Data byte, the device in turn removes the SYN line and the instruction is complete.

5.3.5 Sense Status SS



The Sense Status instruction transfers an eight bit Status Byte from the addressed device controller to the Processor. Status bits take on different meanings for different device controllers. In general, however, data may only be transferred between the Processor and the device when the status of that device is all zeros. Figure 5-9 provides the Device Controller Logic for the Sense Status Instruction.

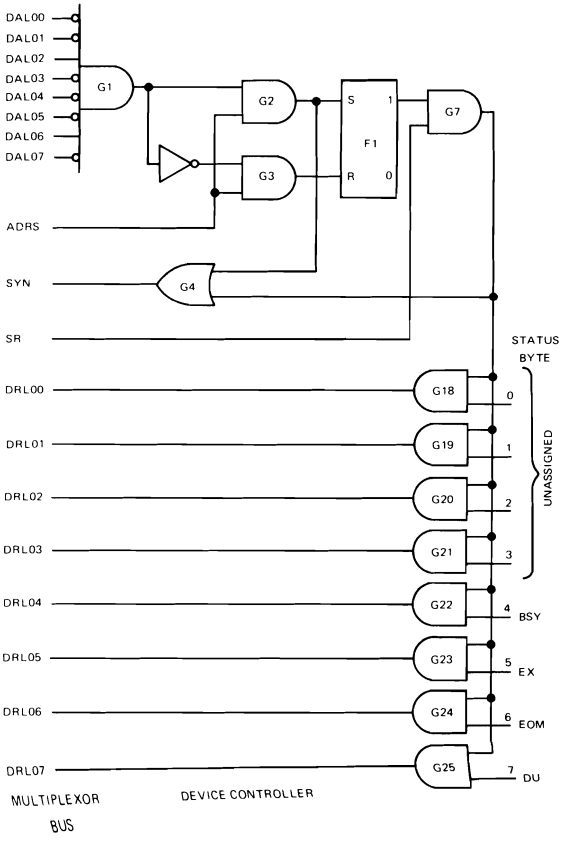


Figure 5-9. Device Controller Logic for Sense Status Instruction

1. The Processor actuates the Status Request (SR) control line.
2. The currently addressed device controller places an eight-bit Status Byte on the Data Request Line DRL00:07 via Gates G18 through G25 when enabled by gate G7.
3. The output from G7 also sends SYN to the Processor through Gate G-4.
4. When the Processor receives SYN it loads the Status Byte from the Data Request Line into the Accumulator. The Processor then releases the SR control line.
5. When the Processor releases the SR control line, the device controller should remove the Status Byte from DRL00-07 and it should also release SYN, and the instruction is complete.

5.3.6 Read Data RD

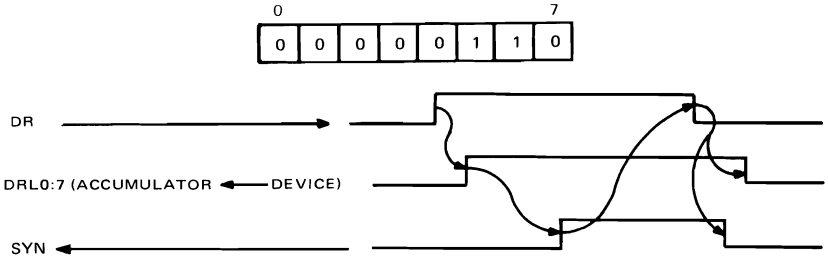


Figure 5-10 provides the device controller logic for the Read Data Instruction.

1. The Processor activates the Data Request (DR) control line.
2. The currently addressed device controller places an eight bit Data Byte on DRL00:07 via Gates G10-G17 when enabled by Gate G5.
3. The output from gate G5 also causes the device controller to send SYN to the Processor via Gate 4.
4. When the Processor receives SYN it loads the Data Byte from the Data Request Lines to the Accumulator. The Processor then releases the DR control line.
5. When the Processor releases the DR Control line, the device controller removes the Data Byte from DRL00-07 and SYN line, and the instruction is complete.

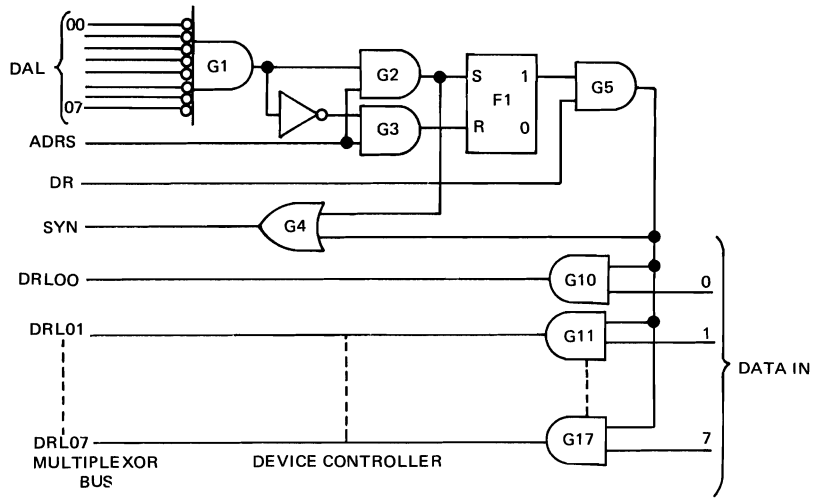
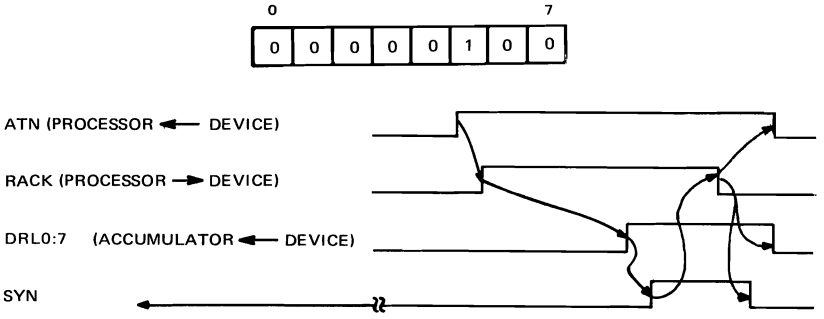


Figure 5-10. Device Controller Logic for the Read Data Instruction

5.3.7 Acknowledge AK



This instruction is used to identify which of up to 256 device controllers interrupted the Processor over the common Attention (ATN) Line.

Figure 5-11 provides the device controller logic for the Acknowledge Interrupt Instruction.

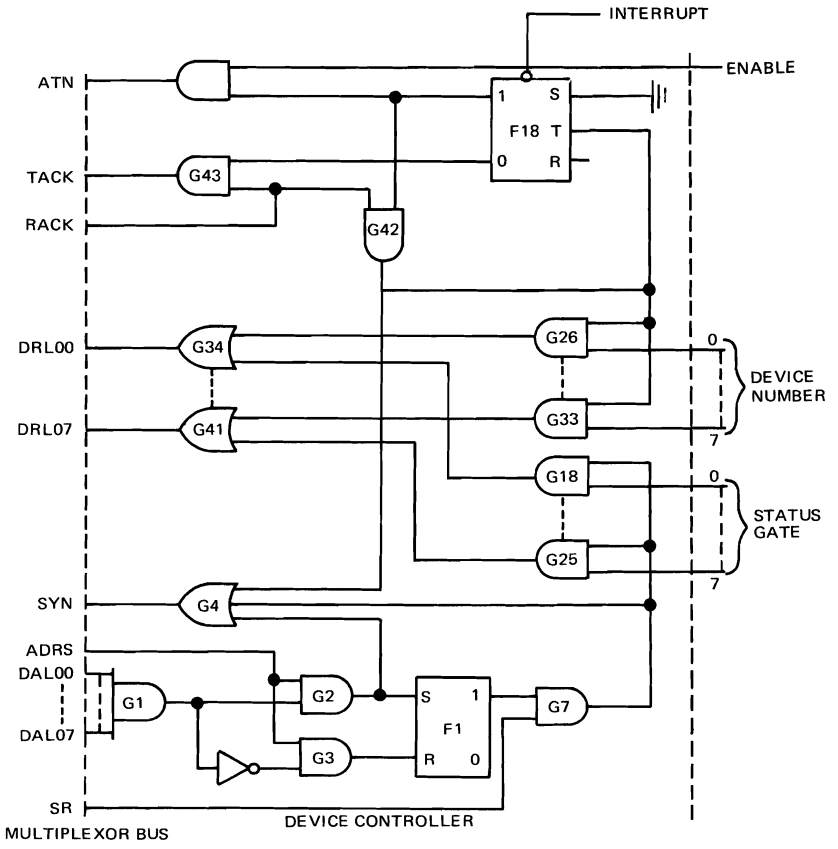


Figure 5-11. Device Controller Logic for the Acknowledge Interrupt Instruction

1. The device controller wishing to interrupt the Processor raises the ATN line by setting flip-flop F18.
2. The program responds by issuing an Acknowledge Instruction.
3. The Processor raises the Acknowledge (ACK) control line.

NOTE:

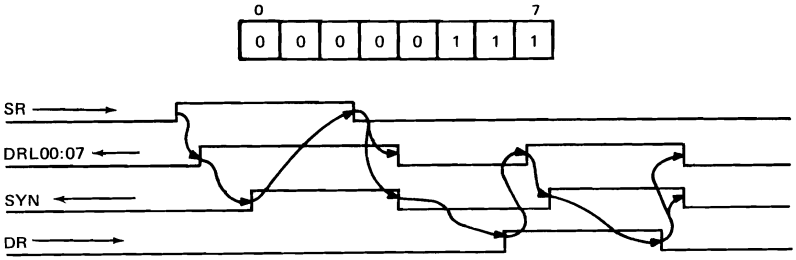
The ACK line is received by the first device controller in the priority daisy chain as Received Acknowledge (RACK). If F18 is not set, the RACK signal is sent on to the next device controller as transmit Acknowledge (TACK) where it is received as RACK, and so on. The ACK line is passed on from device controller to device controller, in order of their priority, until it finds one with the Interrupt flip-flop (F18) set. The Zero output of F18 prevents the RACK pulse from being sent on as TACK.

4. The F18 high output and RACK are ANDed to enable the Device Number from the device to the DRL000 through DRL070 lines, and to send SYN to the Processor to indicate that the device number is on the lines. The ATN flip-flop is also reset.

5. The processor gates the device number into the Accumulator.

6. The Processor then lowers the ACK line which, in turn, causes the device controller to lower the SYN line.

5.3.8 Read Data and Skip RDS



The Read Data and Skip instruction is a combined Sense Status and a Read Data instruction where the Processor tests the device status and, when the device is ready, reads a data byte from the device. See Figure 5-12.

1. The Processor raises the Sense Status (SR) control line. The Status Byte from the currently addressed device controller is returned to the Processor on the DRL line in a fashion similar to that described in the Sense Status instruction.

2. The Processor tests the least significant four bits of the returned Status Byte.

   2a. If any of Bits 5, 6, and 7, (Examine, End of Medium or Device Unavailable), is true, the Processor loads the Accumulator with the Status Byte, and executes the next instruction.

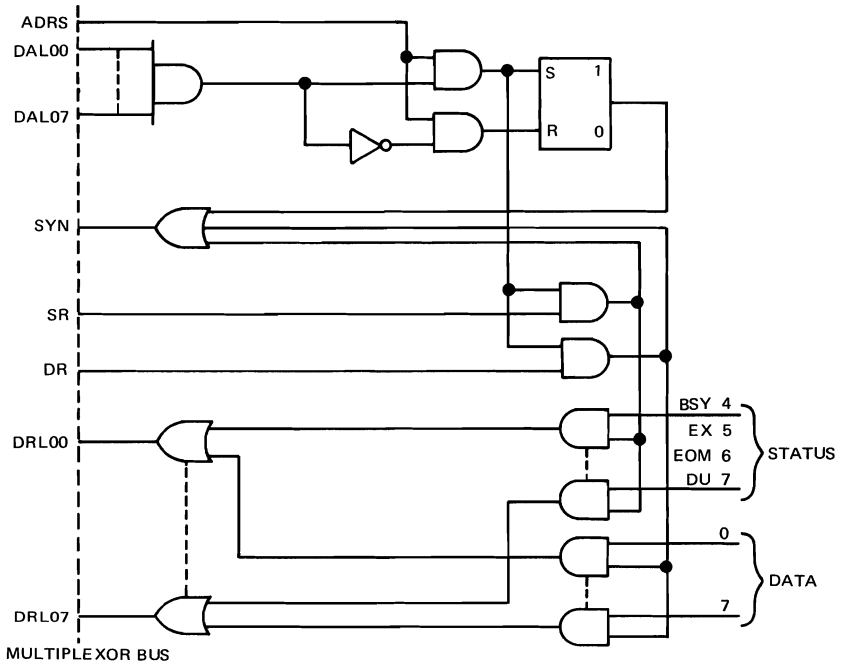


Figure 5-12. Device Controller Logic
(Read Data Skip or Read Block Instruction)

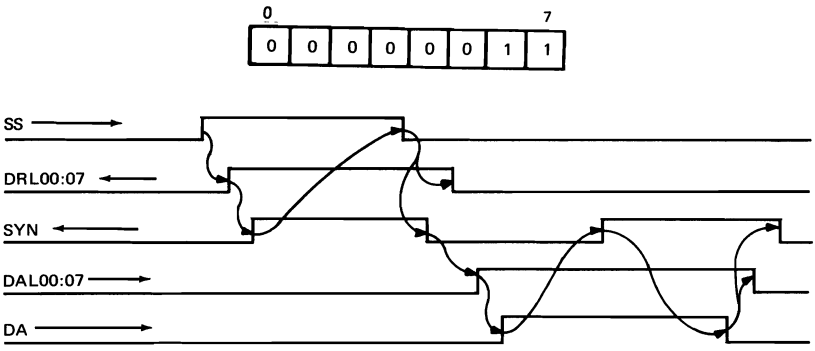
2b. If only Busy is true, the Processor maintains the SR control line active until the Busy Bit becomes false and continues to Step 3.

3. When the device releases the SYN line for the Sense Status portion of the instruction, the Processor activates the Data Request (DR) Control Line.

4. A data byte is transferred to the Accumulator in a fashion identical to that described for RD.

5. On successful execution of this instruction, the next program instruction is skipped.

5.3.9 Write Data and Skip WDS



The WDS instruction is a combined Sense Status and Write Data instruction where the Processor tests the device status and, when the device is ready, sends a data byte to the device. See Figure 5-13.

1. The Processor raises the Sense Status (SR) control line. The Status Byte from the currently addressed device controller is returned to the Processor on the DRL line in a fashion similar to that described in the Sense Status instruction.

2. The Processor tests the least significant four bits of the returned Status Byte.

   2a. If any of bits 5, 6, and 7, (Examine, End of Medium, or Device Unavailable), is true, the Processor loads the Accumulator with the Status Byte, and executes the next instruction.

   2b. If only Busy is true, the Processor maintains the SR control line until all status bits are false and continues to Step 3.

3. When the device releases the SYN line for the Sense Status portion of the instruction, the Processor places the Accumulator on DAL0:7 and raises the Data Available (DA) control line.

4. The device controller receives the Data Byte as described in the WD instruction.

5. A successful execution of this instruction results in the next program instruction being skipped.

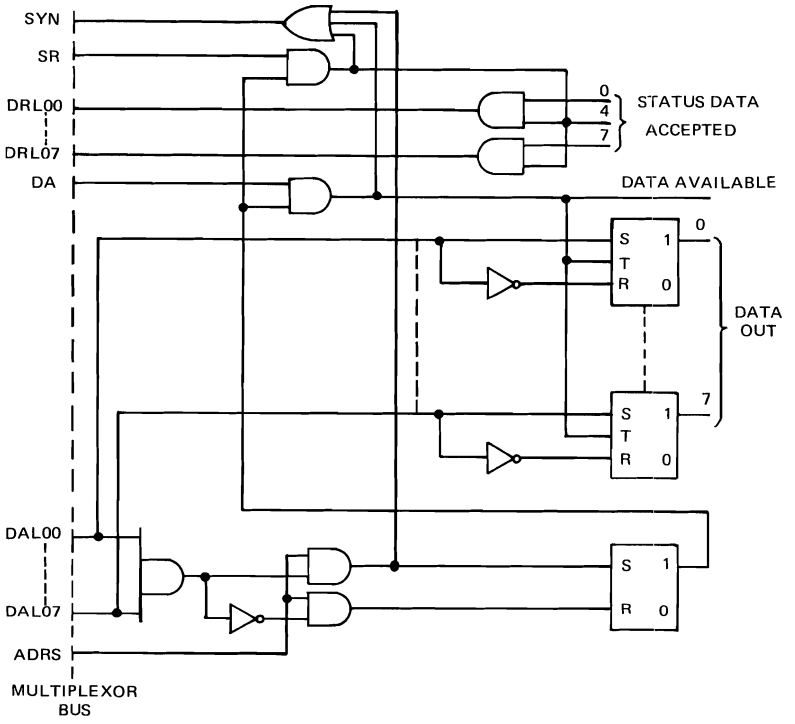
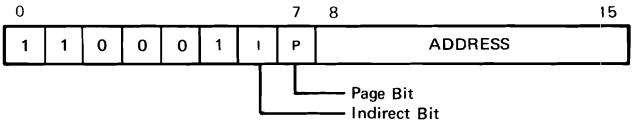


Figure 13. Device Controller Logic
(Write Data and Skip and Write Block Instruction)

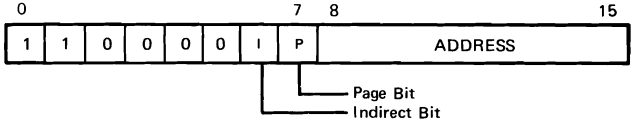
5.3.10 Read Block RB



The Read Block instruction is an extension of the Read Data Skip instruction where a block of up to 256 data bytes is read from the device to memory.

The steps in this instruction are the same as RDS except the operation is repeated until the end of a memory page is reached.

5.3.11 Write Block WB



The Write Block instruction is an extension of the Read Data Skip instruction where a block of up to 256 data bytes is read from the device to memory.

The steps in this instruction are the same as RDS except the operation is repeated until the end of a memory page is reached.

5.4 DEVICE CONTROLLER LOGIC DESIGN

This section describes the procedure to follow in designing I/O device controllers. While it would be impossible to describe all possible controllers, representative circuits are described in enough detail to permit design of most controllers. Note that a General Purpose Interface Controller which simplifies device controller design is described in a later section.

5.4.1 Multiplexor Channel

The I/O Bus system (either Multiplexor or Selector Channel Bus) consists of 27 shared, unidirectional leads which may be divided into four groups:

1. Data Available Lines (DALs) form a group of eight lines from the Processor and carry Address, Command, or Data Bytes to the device controller.

2. Data Request Lines (DRLs) form a group of eight lines which carry status, device address, or data bytes from the device controller to the Processor. In the Processor, the lines are gated into the A Register.

3. Control Lines (CLs) form a group of eight lines from the Processor. Control Lines are raised on a one-out-of-eight basis. These lines define the use and intent of the DALs and DRLs. One of these lines, CL050, carries the Interrupt Acknowledge (ACK) signal and is not a shared line, but breaks up into a series of short lines to form the daisy-chain priority system. The device controller closest to the Processor has the highest priority since the ACK signal must pass through it first.

4. System Synchronize (SYNC), Interrupt Attention (ATN), and System Clear (SCLR) lines form the last group. The SYN and ATN lines carry signals to the Processor where they are used in the timing and control of the I/O Bus system. A SYN signal indicates that the device controller circuit has received a signal on one of the control Lines. The ATN line is raised when any of the device controller circuits cause an interrupt. Access to the ATN line is under control of an Enable (EBL) flip-flop in each device controller. The SCLR0 line provides a relay contact closure to ground which is used to set up initial or preferred states in each device controller.

All buses are of the false type, i.e. zero active. The device controller circuits used to communicate with the I/O Bus system are shown in Figure 5-14. In a typical case, the DALs and Control Lines are buffered by standard gates to drive the Address, Command, Control, and ATN/ACK circuits. The signals back to the Processor on the DRLs are gated by open collector power gates whose outputs are OR tied within the device controller and on the bus. The load resistors for the DRLs are located in the Processor. The paragraphs which follow list the conditions affecting bus usage, and provide a set of design rules. Standard circuits for ATN/ACK and address decoding are also shown.

The System Interface uses Diode-Transistor Logic (DTL) power gates for bus drivers on the I/O Bus system. On the DAL and Control Lines, the line drivers located in the Processor are capable of handling 25 DTL loads in addition to a 1K pullup resistor. The ATN0, SYN0, and DRL Bus Lines are driven by power gates distributed throughout the device controller boards.

On each line, the gate collectors are OR tied and share a common load resistor (located in the Processor) as shown in Figure 5-14. The value of the load resistor, and the number of OR ties, is determined by the total OFF leakage current of the power gates, considered with the maximum ON current of a single gate whose saturation voltage is still below the logical ZERO level.

Calculations for the bus load resistor in Figure 5-14 and for the allowable fan-in, show that for worst case conditions and a 0.7 volt noise margin, the fan-in is about 50 gates. The basic rules for device controllers tying to the I/O Buses are:

1. Only one DTL load should be placed on each device controller input line from an I/O Bus (DAL Control Lines, and SCLR0 line).

2. Not more than two DTL power gates should be OR tied to a device controller output line to an I/O Bus (ATN, SYN, and DRLs).

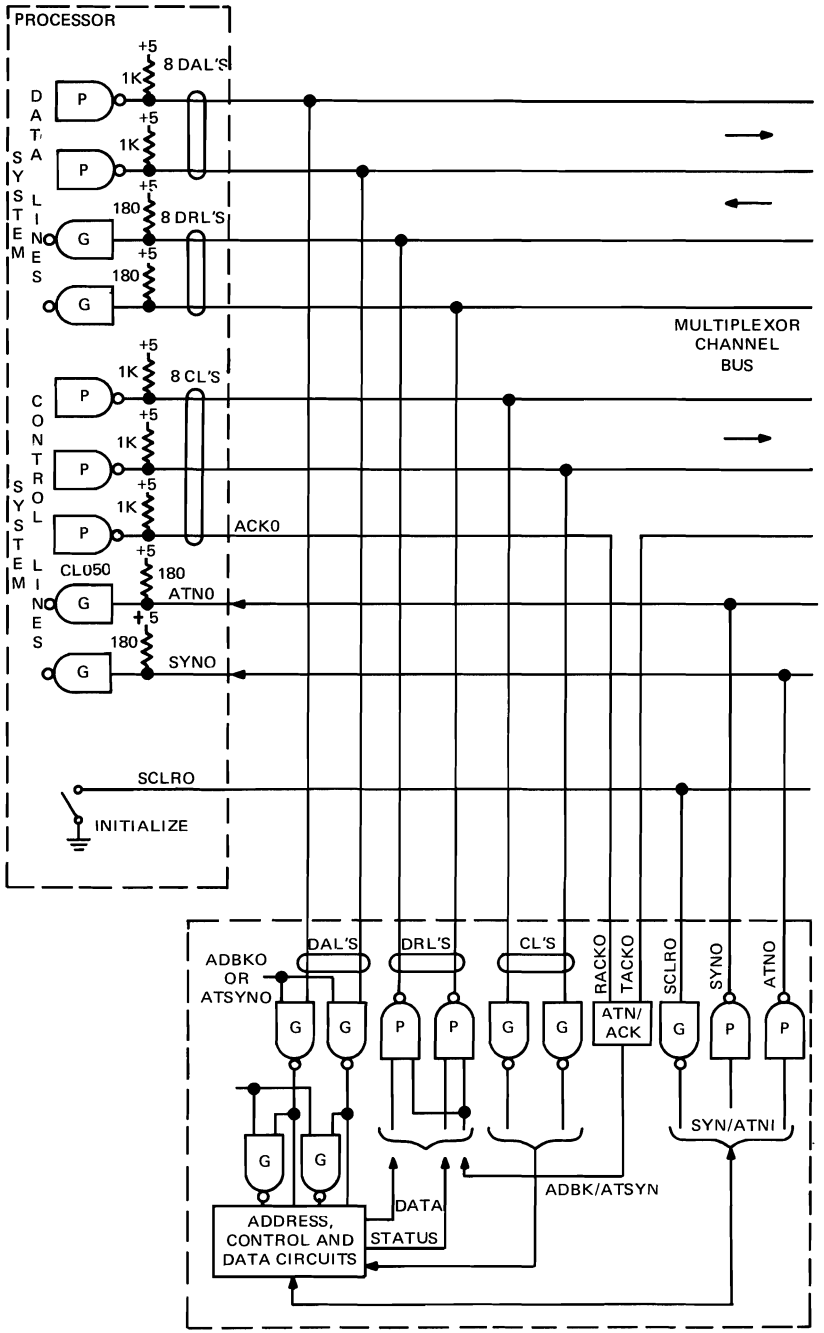


Figure 5-14. I/O Bus Communication Circuits, Logic Diagram

The previous two rules give the Processor a Basic I/O drive capability of 25 device controllers. Additional buffering of the Multiplexor Bus may be achieved, if required, as shown in Figure 5-15.

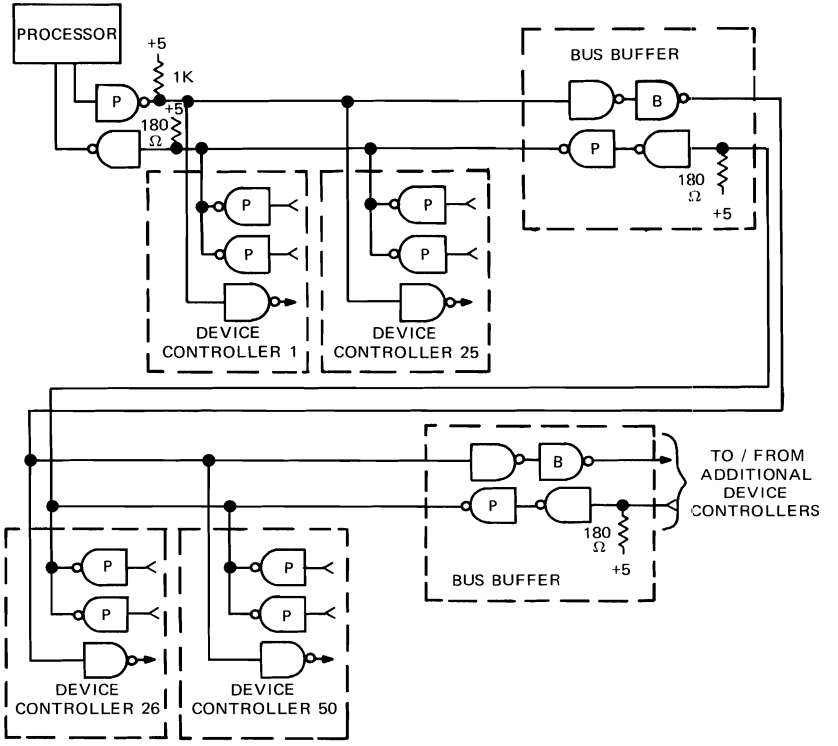


Figure 5-15. Multiplexor Channel Bus Buffers

5.4.2 Device Controller Addressing

Refer to Figure 5-16 during the following description. The dotted lines around the groups of logic functions represent INTERDATA standard logic. Further details on the logic packs may be found in the INTERDATA Logic Module Handbook, Publication Number 29-005. The designer may use his own logic packs provided they are level compatible with INTERDATA logic as described in the aformentioned publication. When a device controller is addressed, the eight-bit address

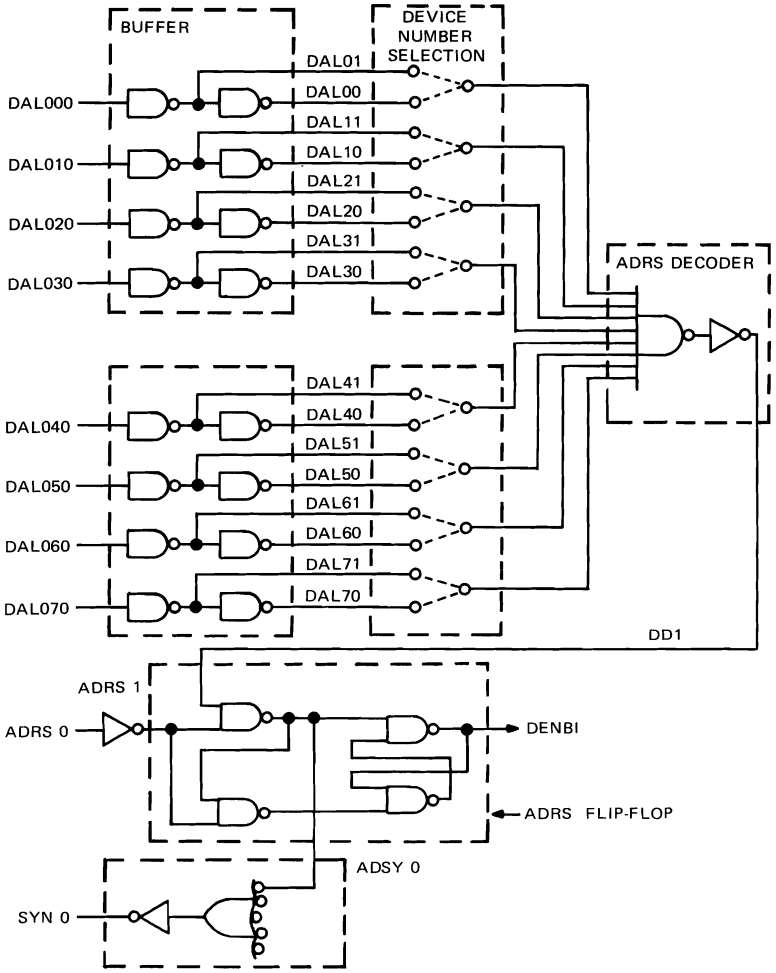


Figure 5-16. Device Addressing, Logic Diagram

code is placed on the Data Available Lines (DAL000 through DAL070). The two buffers provide the true and false DAL lines. The Address Decoder circuit is hard-wired on each controller with its assigned address code, and the eight coded outputs are applied to an eight input gate. Thus, the Decoded Device output (DD1) goes true. The Address control line, ADRS1 then strobes the DD1 line into the ADRS flip-flop.

The Synchronize signal is returned to the Processor, during the presence of ADRS1, via the Address Sync line, ADSY0. Notice that an OR gate is used here for returning the other device Command Sync lines. The set output from the Address flip-flop, called Device ENable (DENB1), is used to gate all other I/O control lines to the device controller. When another device is addressed, the decoded device line, DD1, is low, causing the ADRS1 strobe line to reset the Address flip-flop, and disabling the controller. Thus, only one device controller may be addressed at any time. During the address cycle, only the device that was addressed returns a Sync.

NOTE

The designer must design the Controller such that when some other device is addressed, the previously addressed Controller will clear its address flip-flop within 350 nanoseconds. Otherwise the system could have two devices addressed simultaneously.

The device controller logic must delay sync until it has reacted to the Multiplexor Bus Control Line, however, unnecessarily long delays serve only to reduce the system throughput.

5.4.3 Data and Status Input

Figure 5-17 shows how a byte of data and status may be read into the Processor. When the device is addressed, DENB1 is high, enabling the Status Request (SR) or Data Request (DR) control line. The SR or DR in turn, enables the Status or Data Bytes onto the Data Request Lines (DRL000 through DRL070). Open collector power gates are used for OR typing multiple data sources onto the DRL lines. A system requirement is that the addressed controller must respond to all control lines with a SYNC. The device controller logic should place a high on BSY1 until the data is ready and settled on the Data Request Lines (DR010 through DR070). The Processor may now be synchronized to the device data rate by testing the device status until the Busy Bit is low. Then, when the Busy Bit is low, the program may transfer data. Device synchronization can also be achieved by generating an interrupt when the data is ready.

The End Of Medium (EOM) Bit is normally placed high at the termination of the device medium, such as End of Card. The Device Unavailable (DU) Bit typically signifies that device power is not turned on.

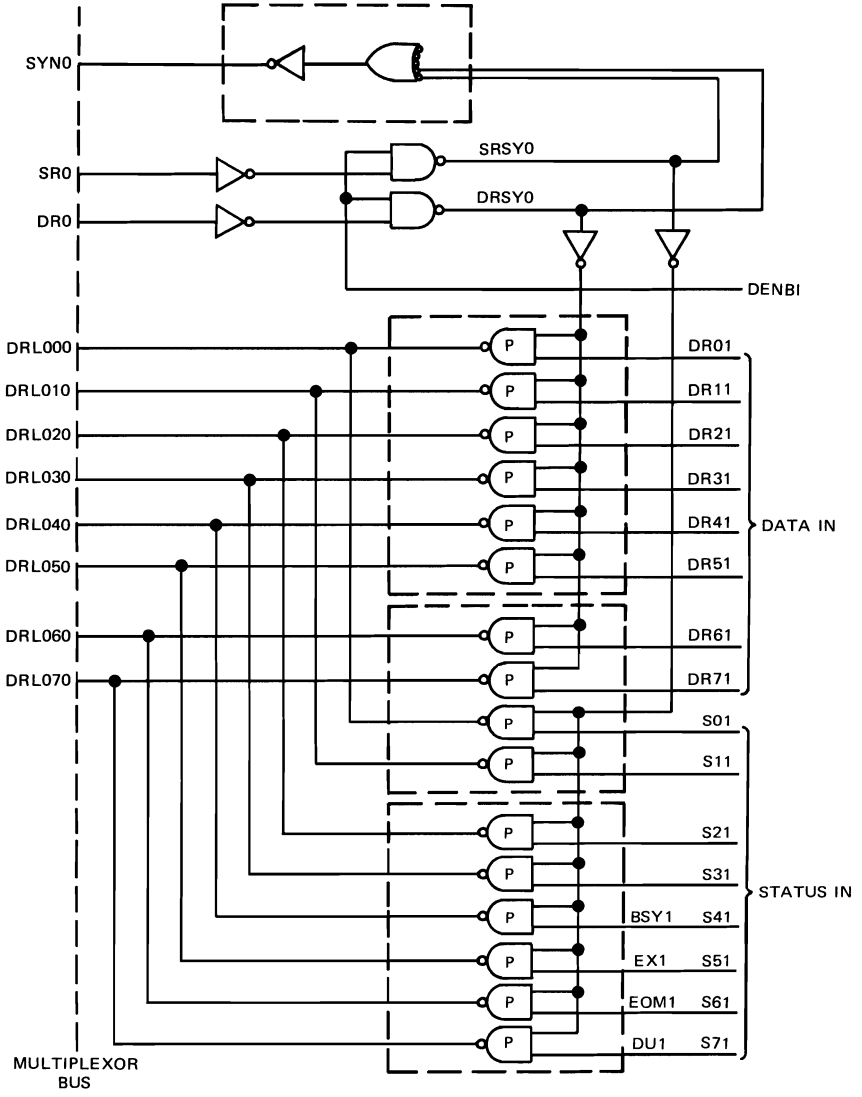


Figure 5-17. Data and Status Input, Logic Diagram

The Examine Status (EX) Bit is used to signify other appropriate device conditions. In this case, the user assigns S01 through S31 to appropriate conditions, such as Parity Error, etc.

It is appropriate to note here that the Busy Status is unconditionally defined such that data cannot be transferred unless Busy is false. The remaining Status Bits are defined as required by the device controller. Not all device controllers require all eight Status Bits.

Device controllers must be designed such that the Processor or the Selector Channel maintains the Status Request line once the current status of the device is presented. Specifically, if the status changes while the Status Request line is true, the status byte returned to the Processor or Selector Channel should also change.

5.4.4 Data and Command Output

Figure 5-18 shows how a byte of data and command may be output from the Processor. The buffered true and false Data Available Lines, DAL001 through DAL071 and DAL000 through DAL070 from Figure 5-16 feed to the set and reset inputs of the data Register. When the device is addressed, DENB1 is high, enabling the control line DAG1 to strobe the data condition into the J-K flip-flop Data Register. The DASY0 line also returns the Sync signal to the Processor.

The Command Lines are shown on Figure 5-18 as being used in the Toggle Mode. For example, a high on Bit 0 (DAL001) sets a control relay when CMG1 goes high. A high on Bit 1 (DAL011) resets the relay. Bits 6 and 7 are shown operating an indicator. Other pairs of bits may be used to enable/disable interrupts, etc.

Again, note that definition of the Command Bits is a function of the device controller only. Not all device controllers require eight separate commands. However, up to 256 commands are possible.

5.4.5 Interrupt Control

Figure 5-19 shows a complete general purpose interrupt and interrupt acknowledge logic system. When an interrupt is generated, the Queue flip-flop is DC set via a differentiated negative going pulse. The output from the Queue flip-flop generates an Attention signal (ATN0) to the Processor. ATN0 is connected to one of the four standard priority interrupt lines in the Processor. The Processor responds with an Acknowledge Interrupt signal which is received by the controller as Receive Acknowledge (RACK). Since the queue flip-flop was set prior to receiving the RACK, the Gate G1 output disables G9, holding the G9 output high. The high output from G9 stops TACK0 from sending the Acknowledge to the next device. Thus, RACK1 and the G2 output generate ATSY0 via G3. ATSY0 sends a SYNC back to the Processor, and also forces all inputs (DAL000 through DAL070) to zero. This

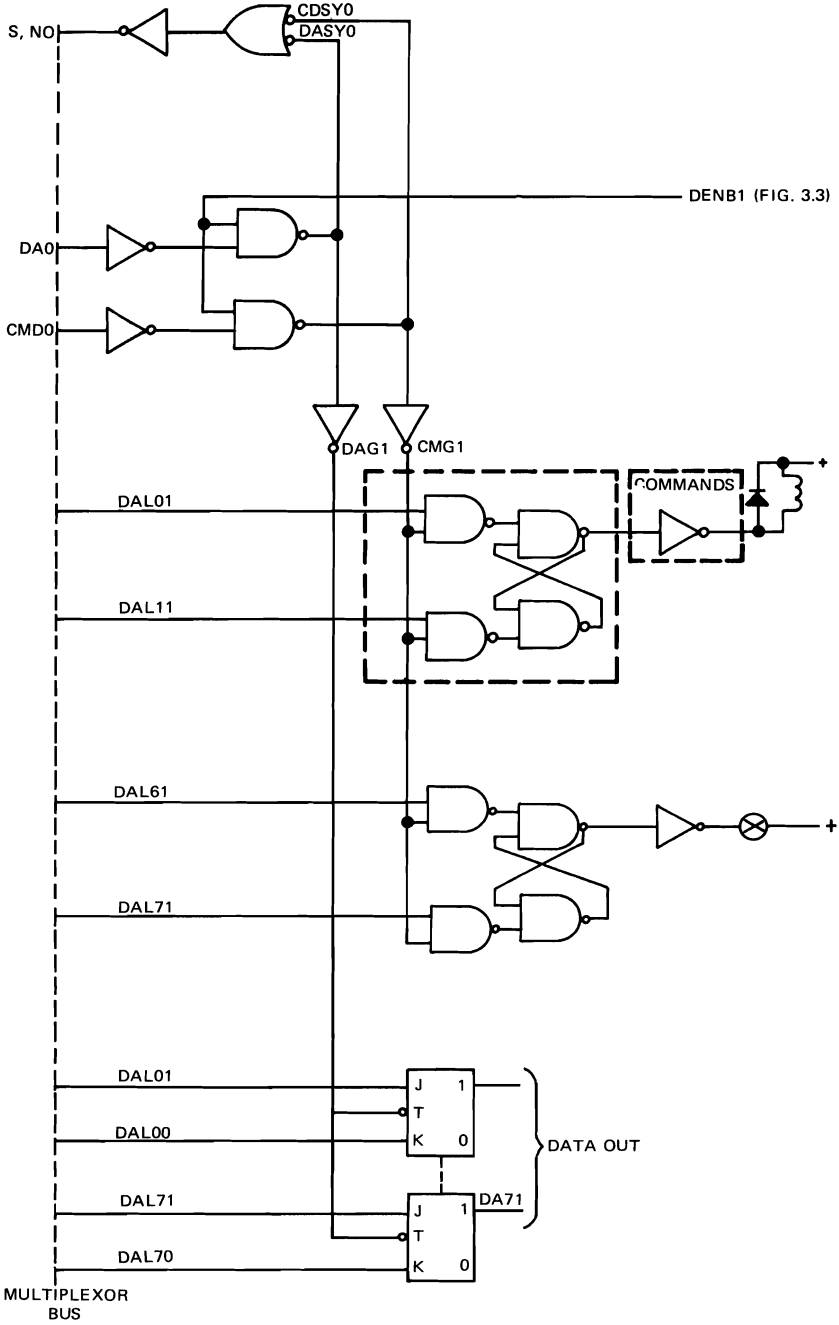


Figure 5-18. Data and Command Output, Logic Diagram

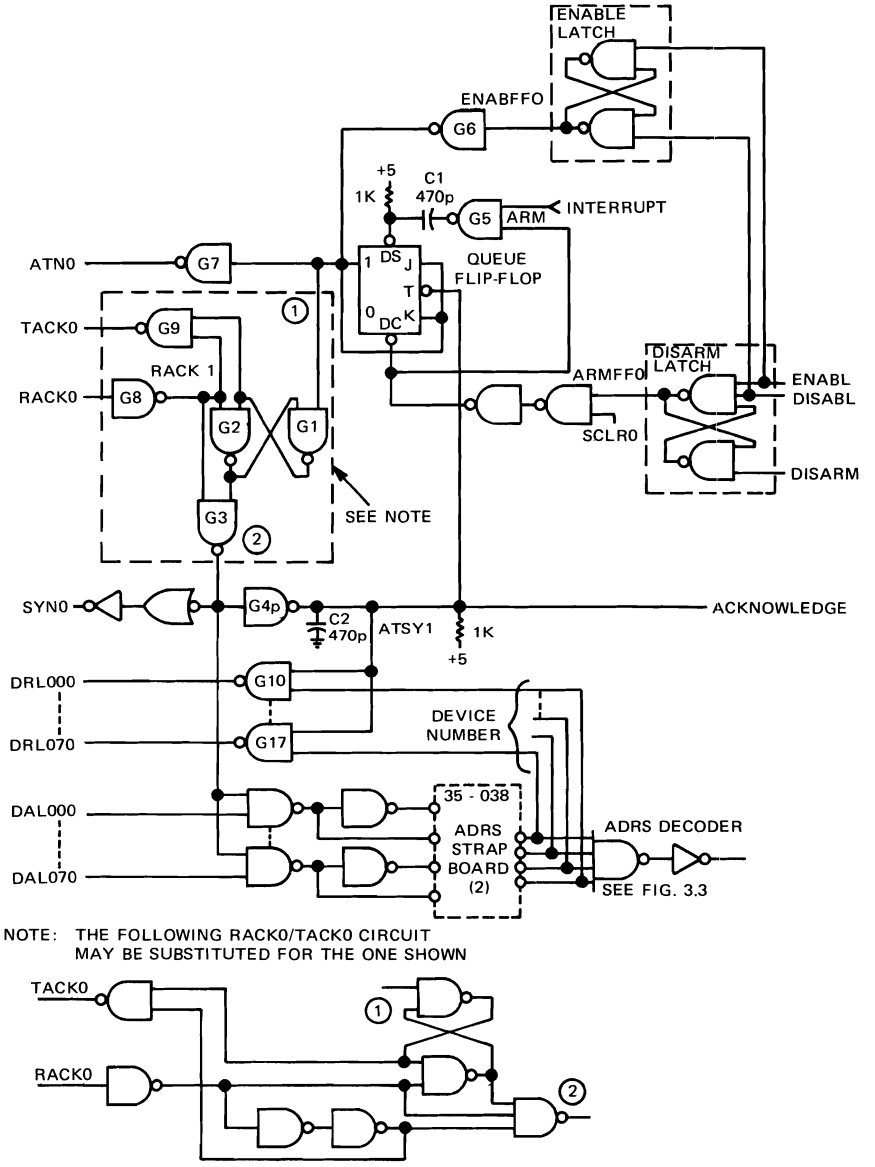


Figure 5-19. Interrupt Control, Logic Diagram

causes the device number wired in by the address strap board to appear on the inputs of G10 through G17. Thus, the ATSY1 output from G4 enables the device number onto DRL000 through DRL070.

Capacitor C2 removes a 30 nanosecond false pulse which appears if the Queue flip-flop is set at the same instant that RACK0 is received in response to another device interrupt. This pulse might otherwise reset the Queue flip-flop before the interrupt is serviced. Note that an alternative method to prevent the occurrence of the false pulse is indicated in the note on Figure 5-19. Here C2 is not required. The output from G4 also raises the Acknowledge signal to the device. On receiving the SYN0, the Processor lowers RACK1, causing the output of G4 to drop. This in turn causes the Queue flip-flop to reset.

NOTE

If the interrupt has not set the Queue flip-flop, the RACK1 signal passes through G2 to TACK0, and on to the next device.

If RACK1 is high in response to another device, the output from G2 is low, thus disabling the interrupt from affecting G1. However, the interrupt remains in the Queue flip-flop, and is serviced after completion of the previous interrupt service.

The ENABFFO and ARMFFO lines provide control over the Interrupt Queue flop-flip and the ATNO line to the Processor. Normally, two bits of a command byte (Bits 0 and 1) are decoded such that, with Bit 0 true and Bit 1 false, the queue flip-flop is disabled. That is, the flip-flop may be set, however, its output is held low. G6, whose input is ENABFFO from the false side of the ENABLE latch, provides the function. The command byte, with Bit 0 false and Bit 1 true, is decoded (ENABL goes false) and sets the ENABLE latch which allows new interrupts or a queued interrupt to be recognized. Bits 0 and 1, both true, are decoded to drive DISARM false which sets the DISARM latch. The false side of the latch is used to clear the Queue flip-flop and to prevent the Interrupt line from setting it. The DISARM latch is cleared whenever the ENABLE or DISABLE commands are recognized. Encoded commands ENABLE/DISABLE/DISARM thus provide interrupt masking or inhibiting within the device controller.

As described previously, the Control Line, CL050, from the Processor carries the Interrupt Acknowledge (ACK) signal. This line breaks up into a series of short lines to form the daisy-chain priority system. The ACK signal must pass through every controller that is equipped with Interrupt Control circuits. This includes all device controllers except a few special cases.

Back Panel wiring for interrupt control is shown in Figure 5-20. At a given position, the Received ACK (RACK0) appears at Pin 114-0 and

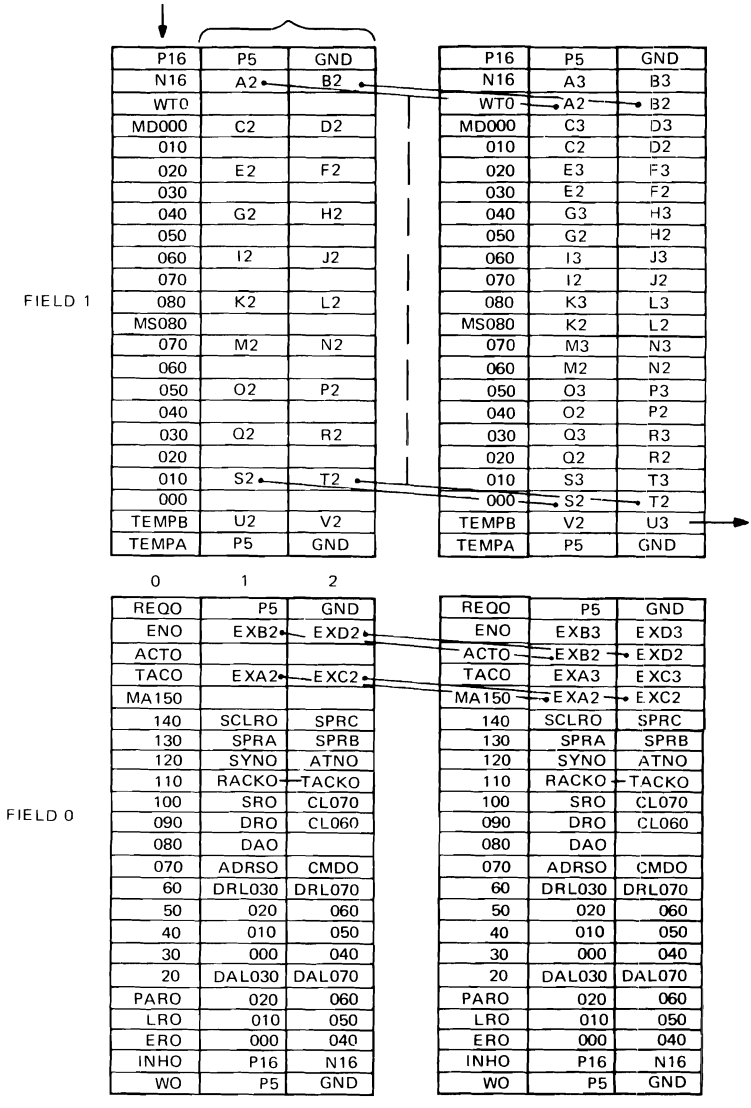


Figure 5-20. Typical Universal Expansion Slot Wiring

the transmitted ACK (TACK0) at Pin 2140. The daisy-chain bus is formed by a series of isolated lines which connect Terminal 214-0 of a given position to Terminal 114-0 of the next position (lower priority). On unequipped positions, a jumper shorts 114-0 and 214-0 of the same connector to complete the bus. Back Panels are wired with jumpers on all positions. Whenever a card chassis position is equipped with a controller, the jumper from 114-0 to 214-0 must be removed from the Back Panel at that position.

For controllers that occupy several positions, the jumper is removed only at the position where the controller board has ATN/ACK circuits.

5.4.6 Multiplexor Channel Wiring

Wiring for the Multiplexor Channel and the Selector Channel is identical. The bus connections are via the bottom connector (Field 0) and are shown on Figure 5-20. Note that the top four rows of pins are stitched for use in communicating between mother-boards in the same device controller.

The top connector (Field 1) is stitched as shown. The Field 1 connector is used solely for communication between mother-boards.

5.4.7 Multiplexor Channel Timing

Both the Input and Output operations on the Multiplexor Channel make use of "request-response" signaling. This allows the system to run at its maximum speed whenever possible, but permits a graceful slowdown if the characteristics of a particular device controller require signals of longer duration. Device controler designs should keep Multiplexor Channel usage as fast as possible consistent with practical circuit margins. Doing this assumes the greatest computer throughput when a system is configured with a number of peripheral devices.

Typical operations are shown on Figure 5-21 for Input and Output. On the Output operation, the Processor places a signal on the Data Available Lines followed by an appropriate Control Line signal. This stagger (T1) will vary depending upon which model Processor is in use, but it is guaranteed to be at least 100 nanoseconds. When the device controller has received the Output Byte, the SYN signal is returned to the Processor which then terminates the Control Line signal. Realizing that T5 is 100 nanoseconds minimum, the SYN delay T2 should be only long enough to guarantee proper reception of the Output Byte. The Control Line/DAL removal time (T3) is important where single-rail to double-rail operation is used - e.g. the ADRS flip-flop on Figure 5-16. A minimum of 100 nanoseconds is guaranteed for T3. For SYN generation as per Figure 5-16 and 5-19, the Control Line signal is DC coupled through the gates to form the SYN signal. The SYN removal time (T4) will be the delay through four DTL gates. This delay should not be unnecessarily extended since the Processor will not consider the Input/Output operation complete until SYN0 falls.

Device controllers must be assigned to accept a minimum control pulse width (particularly ADRS) of 300ns.

It should be emphasized that the times shown on Figure 5-21 are defined for signals on the Multiplexor Channel. Within a given controller, one signal may flow through more gates than another signal and these delays must be considered.

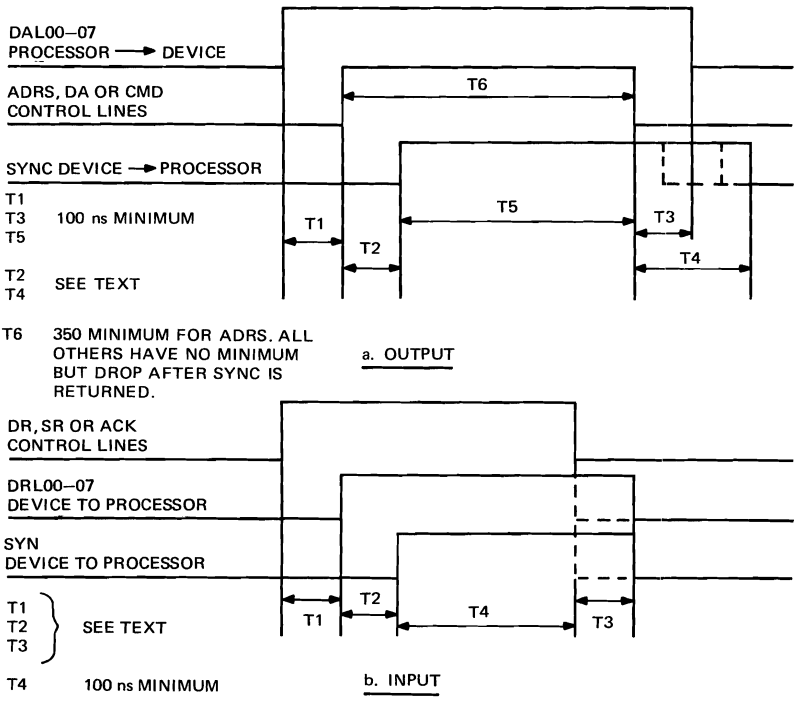


Figure 5-21. Multiplexor Channel Timing

For the Input operation, the Processor places a signal on a Control Line. The currently addressed device controller should gate signals to the DRLs as soon as possible to keep T1 at a minimum. The SYN delay (T2) must guarantee that the Input Byte is on the DRLs considering the slowest data gates and the fastest SYN gates. The Processor will remove the Control Line signal when SYN is received with a minimum delay (T4) of 100 nanoseconds. With SYN and the byte gate DC coupled to the Control Line, the removal delay (T3) will be the sum of the corresponding gate delays. The Processor considers the operation complete when SYN falls.

When the Control signal is ACK, the delay T1 will include the cumulative G8/G9 delays (See Figure 5-21) for all the controllers between the responding controller and the Processor. This will be less than the Processor Time out even with the maximum limit of 256 controllers.

NOTE

It is essential to realize that after the Processor initiates a Control Line signal, the Processor does nothing until the SYN signal is returned by the device controller; one or more cycles are skipped if necessary and the data transfer rates decreased proportionally. While this may not affect a

particular controller, the overall system performance is degraded. Furthermore, if a device controller fails to respond with a sync in the time out period of about 35 microseconds, the Processor will abort the instruction.

5.5 STANDARD I/O BOARD

5.5.1 Introduction

This section describes a basic bus communications board which is available from INTERDATA to facilitate custom controller design by the customer. The Standard I/O Board is essentially a mother-board which contains the bus communications and interrupt circuits used on most device controllers. A more complete design description is given in the Standard I/O Board Instruction Manual, Publication Number 29-041.

The Standard I/O Board has its IC logic mounted directly on the mother-board in a field near the lower connector (Connector 0. This positioning keeps the signal paths to and from the bus as short as possible. See Figures 5-22 and 5-23. The remainder of the board provides 28 connectors for standard INTERDATA daughter-boards which are described in the Logic Module Handbook, Publication Number 29-005. The same Standard I/O Board may be used with either the Multiplexor Channel or the Selector Channel.

5.5.2 Communications Logic

The Address and SYN circuits on Figure 5-24 perform the same functions as those shown on Figure 5-16. The address straps for device number selection are wired in a field at location 20 on the mother-board. The lettered pins also provide the true and false states for an eight-bit byte which can be wired to Data and/or Command Registers in the wire-wrap portion of the board.

Data and Status Bytes returned to the Processor originate in the customer designed wire-wrap portion of the board and share the bus driver circuits as shown in Figure 5-24. A group of pins designated the E Field provide this connection.

The Interrupt Control circuit on Figure 5-24 is logically identical to that on Figure 5-21. The Data Output and Command Output circuit shown on Figure 5-24 is logically identical to that on Figure 5-20.

5.5.3 Wire-Wrap Facilities

The Standard I/O Board provides 28 daughter-board locations, 23 mother-board pull-up resistors (1K to +5 volts), space for eighteen components (resistor, capacitors, or diodes) mounted between pull-up resistors, space for 24 reed relays form 1A, space for eight PC trim potentiometers, and sixteen test points on the outer end of the

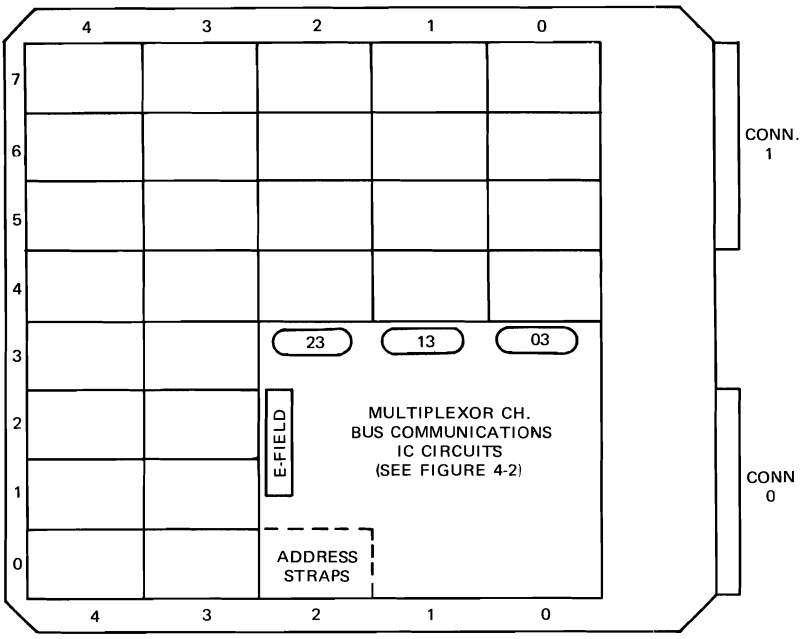


Figure 5-22. Standard I/O Board Layout

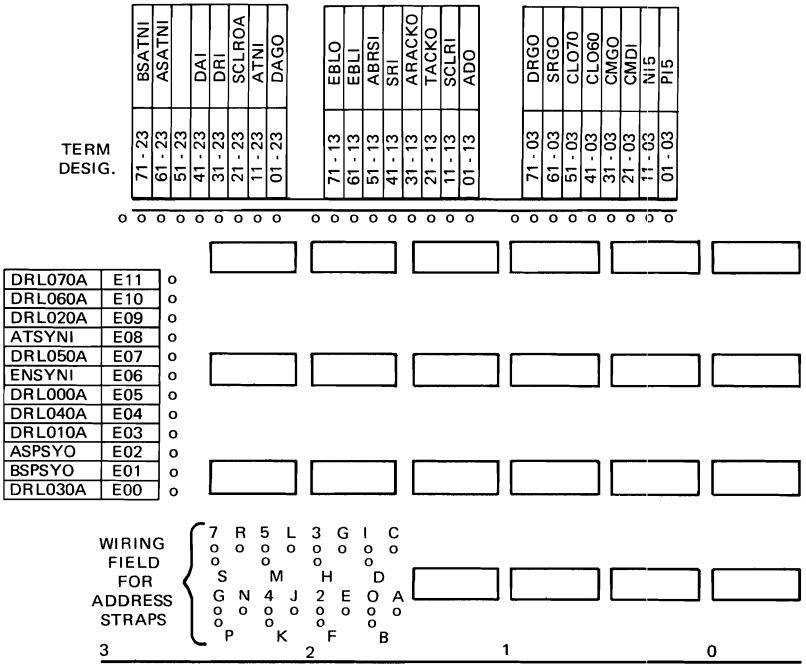


Figure 5-23. Standard I/O Board Field Layout

mother-board. Cable connectors to external devices or circuits are mounted in the daughter-board locations along the outer edge of the board in the same manner as with standard INTERDATA controllers.

Wire-wrap pins for the upper connector pins (Connector 1) are available. These are used when more than one mother-board is required for the controller. The back panel "stitch pattern" shown in Figure 5-23 provides straps between adjacent boards. A multi-board controller usually requires only one Standard I/O Board (35-104) to provide the communications circuits. Wire-wrap mother-boards (35-050) with 40 daughter-board positions each are used for the additional logic.

5.6 PULSED INPUT/OUTPUT

The pulsed mode is the second mode of communicating over the System Multiplexor Bus. Data is transferred to and from device controllers in a synchronous fashion as opposed to the Request/Response (handshaking) associated with the Asynchronous Mode discussed earlier. The same identical Data and Control Lines are shared among device controllers which operate in the two different modes. Device controllers which operate in either mode may be mixed in any way on the Common Multiplexor Bus.

Nearly all standard INTERDATA device controllers operate in the Asynchronous Mode over the Multiplexor Bus. The pulsed I/O Mode is intended principally for custom interface designs to the Model 1 Multiplexor Bus. Custom interface designs using the Pulsed Mode will not be hardware plug compatible with the existing INTERDATA family of Processors (Model 3, Model 4, Model 5, Model 13, Model 14, or Model 15). However, in some cases for unsophisticated interfaces, interfacing to the Model 1 using the Pulsed Mode will render a less expensive and more straight forward interface design, sometimes at the expense of more software overhead.

Device controllers which operate in the Pulsed Mode may interrupt the Processor via the normal I/O Interrupt (ATN) line. If so, they must be designed to respond to the Acknowledge Interrupt instruction as described in an earlier part of this chapter. The custom interface design can be completely independent of the Asynchronous Mode if it interrupts the Processor through some Interrupt Line different from the ATN Line.

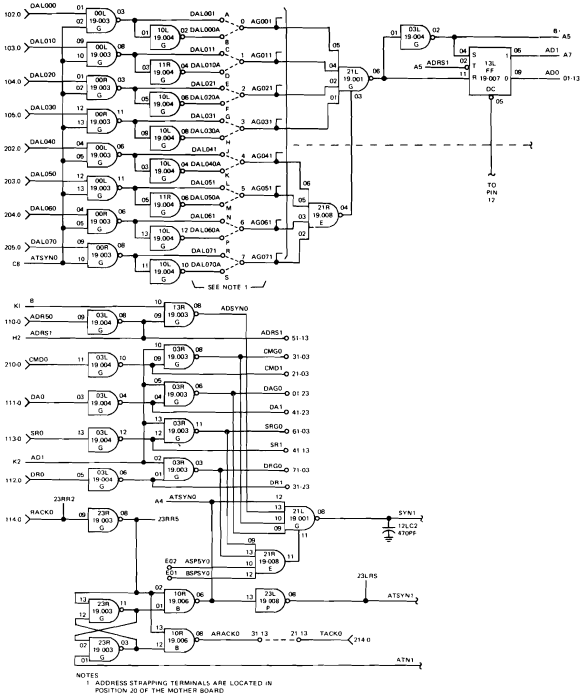
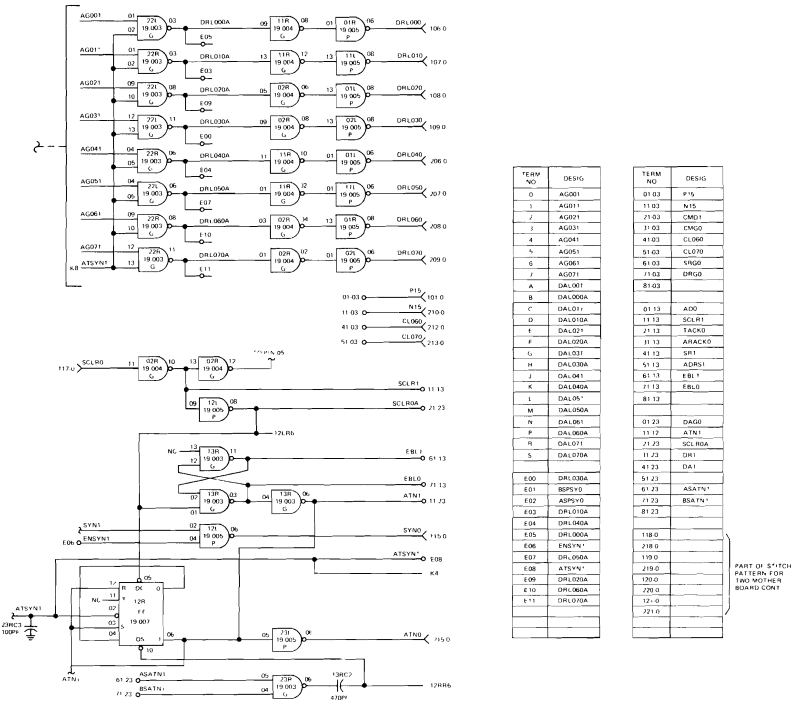


Figure 5-24. Standard I/O Board Schematic
(Sheet 1 of 2)



TERM NO	DESIG
0	AG001
1	AG011
2	AG021
3	AG031
4	AG041
5	AG051
6	AG061
7	AG071
A	DAL001
B	DAL006A
C	DAL011
D	DAL010A
E	DAL021
F	DAL025A
G	DAL031
H	DAL030A
J	DAL041
K	DAL040A
L	DAL051
M	DAL050A
N	DAL061
P	DAL060A
R	DAL071
S	DAL070A
E00	DRL030A
E01	BSPSY0
E02	ASPSY0
E03	DRL010A
E04	DRL040A
E05	DRL000A
E06	ENSYN1
E07	DRL050A
E08	ATSYN1
E09	DAL020A
E10	DRL060A
E11	DRL070A

TERM NO	DESIG
01 03	P15
11 03	N15
21 03	CMD1
11 03	CMD0
41 03	CL060
51 03	CL070
61 03	SNG0
71 03	DRG0
81 03	
01 13	ADD
11 13	SCLR1
21 13	TACK0
11 13	ARACK0
41 13	SR1
51 13	ADRS1
61 13	EBL1
71 13	EBL0
81 13	
01 23	DAG0
11 12	ATN1
21 23	SCLR0A
11 23	DR1
41 23	DA1
51 23	
61 23	ASATN1
71 23	BSATN1
81 23	
118 0	
218 0	
119 0	
219 0	
120 0	
220 0	
121 0	
221 0	

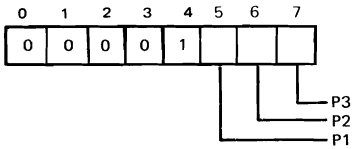
PART OF SWITCH PATTERN FOR TWO MOTHER BOARD CONT

Figure 5-24. Standard I/O Board Schematic
(Sheet 2 of 2)

5.6.1 Pulsed I/O Instruction

The Pulsed I/O instruction takes the form

PULSED I/O PIO



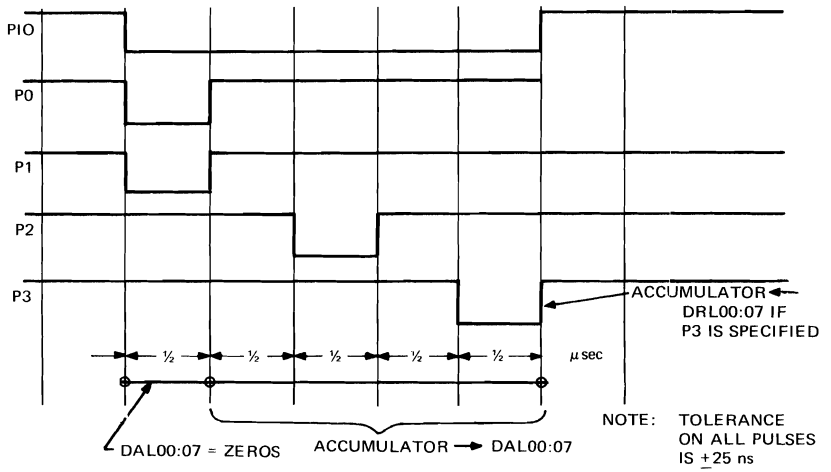
This instruction outputs a control level and a fixed control pulse designated PIO and P0, respectively. In addition, it outputs any combination of the three pulses P1, P2, and P3 as specified by the instruction word. The Accumulator is output over the Eight Data Available Lines except for the interval covered by P0 (at which time the DAL Lines are zeros). The Accumulator is loaded from the eight Data Request Lines (DRL0:7) at the end of P3 if and only if P3 is specified.

The Carry Bit is not changed

Accumulator ← Data Byte (if P3 is specified)

unchanged (if P3 is not specified)

System Configurations which use the Pulsed I/O Instruction should not assign address X'00' to any standard INTERDATA device controller. The Pulsed I/O Instruction will reset the address flip-flop of all standard INTERDATA device controllers connected to the Multiplexor Bus at the end of the P0 pulse (by attempting to address device number zero).



5.6.2 Use of the Pulsed I/O Instruction

The Pulsed I/O Instruction has a wide variety of applications suitable for very simple to very complex device interfaces. There are no restrictions on the definition of the fixed level, PIO, the fixed pulse, P0, or the 3 programmable pulses P1, P2, and P3. Their meaning is defined by the System Designer who uses the instruction. The one restriction placed on the use of the Pulsed I/O instruction is that the system must not include a standard Asynchronous Device Controller whose address is X'00'. This is done so that the PIO instruction can automatically clear the Address flip flop of all Asynchronous Controllers on the Multiplexor. The remaining 255 device controller addresses are legal.

Up to 255 devices may operate in the Pulsed Mode. The definition of the addressing scheme for Pulsed I/O devices is left to the System Designer. In General, the addressing scheme is a function of the number of interfaces in a particular system. Any of 3 addressing techniques could be used.

1. If there is only one Pulsed I/O interface then:

   The PIO level could be used as an address line to enable the custom Pulsed I/O interface.

2. If there are a maximum of 8 Pulsed I/O interfaces, then:

   Each interface may be addressed by individual bits in the Accumulator on a one of eight basis along with the PIO level. The interface may need an address flip-flop or it may not need an address flip-flop. The decision is left to the System Designer.

3. If there are more than 8 Pulsed I/O interfaces, then:

   each may be addressed by encoding any set of the eight data bits into address. Up to 256 addresses may be encoded. Again, as in 2 above, the device may or may not require an address flip-flop.

The three programmable pulses, P1, P2, and P3, may have any meaning the System Designer chooses to assign them. In fact, each pulse can be assigned a different meaning for different interfaces. In general, the programmable pulses could be used in any of the following ways;

1. To identify the meaning of the data from the Accumulator on DAL00:07
2. To steer the data byte on DAL00:07 to different registers.
3. To set up any number of control flip-flops.

Thus using the pulse bits for timing, and the accumulator bits for control, one can use the computer for the timing and control of the device itself, thus saving on the hardware required to control the device at the expense of additional software support.

The fixed pulse P0, occurs at the beginning of the PIO instruction and is not programmable. The fixed level PIO covers the entire interval of the Pulsed I/O instruction and is not programmable. The System Designer is free to assign whatever meaning he chooses to each of these lines.

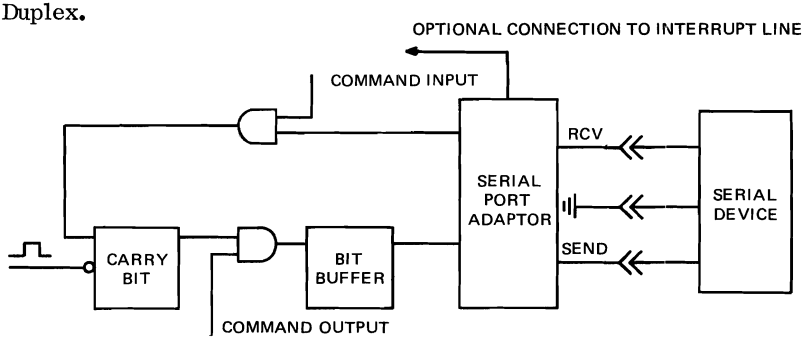
When the pulse P3 is specified, the Processor will load the Accumulator from the eight Data Request (DRL0:7) lines. The DRL lines must be settled within 350 nanoseconds from the beginning of P3 and must remain settled until the end of P3 to insure loading the proper data byte.

The control pulse lines associated with the Pulsed I/O instructions are OR tied onto the standard Multiplexor Bus Control Lines as follows:

PIO	-	CL060	-	212-0	Expansion Slot Pin Numbers.
PO	-	ADRS	-	110-0	
P1	-	DAO	-	111-0	
P2	-	SRO	-	113-0	
P3	-	DRO	-	112-0	

5.7 SERIAL INPUT/OUTPUT PORT

The Serial I/O Port is a built-in feature of the Model 1 Processor which is used for interfacing to low speed bit serial data streams such as those associated with teletypewriters. Data is transferred between the bit serial device and the Carry Bit through the Serial Port at program command. The Serial Port is programmed in the Interrupt Mode using either the built-in One Millisecond Clock or a customer provided clock connected to any available External Interrupt Line. On input operations, the program is required to assemble the serial data stream into characters for stroage in memory. On output operations, the program disassembles characters into a serial data stream. The mode of operation through the Serial I/O Port can be Simplex, Half-Duplex, or Full-Duplex.

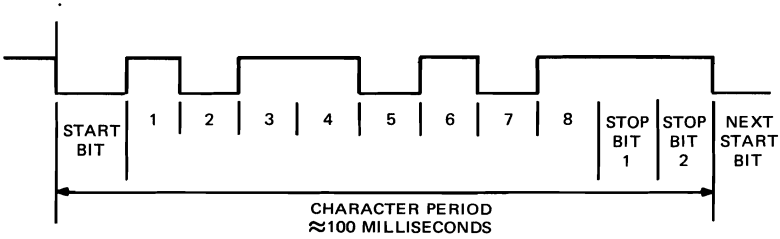


The essential elements of the Serial Port are shown above. The Serial Port Adapter is a plug replaceable circuit (daughter board) which contains the bit buffer and which converts the serial data stream from/to standard DTL/TTL logic levels to levels compatible with the serial device. The Received serial data stream is loaded to the Carry Bit on a program Command. The Bit Buffer is loaded from the Carry Bit on a program Command. The state of the Bit Buffer appears on the Send Line of the Serial Port.

The Serial Port Adapter supplied with the Processor is a daughter board which is designed for a Teletypewriter (Teletype Corporation PD 106 C) with the 20 milliamp option installed. If the user wishes, he may connect a different type of serial device to the Port. Doing so may, depending on the device, require replacing the pluggable adapter with a different circuit.

5.7.1 Operation With a Teletypewriter

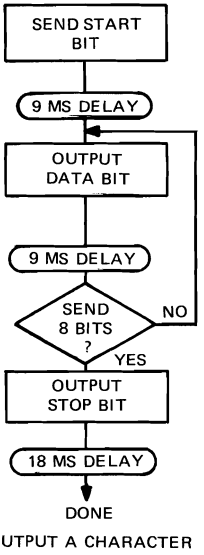
The bit rate from a Teletypewriter is one bit per 9.09 milliseconds. Each character is 11 bits long, including 1 Start Bit, 8 Data Bits, and 2 Stop Bits. Accordingly a character period is 99.99 milliseconds or 100 milliseconds approximately.



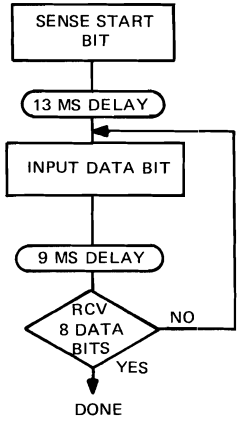
The Start Bit corresponds to a binary one. The Stop Bits correspond to binary zeros. The data bits correspond to eight level ACSII code. The Start Bit and the two Stop Bits have no significant data content. They are used only to identify the beginning and the end of a character period.

TO OUTPUT A CHARACTER TO THE TELETYPEWRITER

The program must output a Start Bit, eight Data Bits, and two Stop Bits, in that order, using the built-in One Millisecond Clock to establish the 9 millisecond bit period. The program disassembles the character by shifting each bit into the Carry Bit and outputting it to the Bit Buffer at the 9 millisecond rate.



OUTPUT A CHARACTER



INPUT A CHARACTER

TO INPUT A CHARACTER FROM THE TELETYPEWRITER

After the program senses a Start Bit, it inputs each data bit at the approximate center of each data bit period. The program assembles the 8 data bits into a character using the Shift instruction. Note that the center of the first data bit occurs about 13 milliseconds after the beginning of the Start Bit (Start Bit period plus one-half a Bit Period). The center of each bit following occurs approximately every 9 milliseconds thereafter.

There are two ways to sense the START Bit on Input operations.

1. The program can test the state of the input line between characters every one millisecond. A binary one corresponds to an idle condition. The first binary zero sensed corresponds to the Start Bit.

2. The user may connect the input line to an available Interrupt Line. The Interrupt Line would be enabled by its mask bit at the end of the last character. The Processor would be interrupted on receipt of the next Start Bit. After this interrupt the program would disable that Interrupt Line until the end of the character period. The procedure for connecting this Interrupt Line is covered in Chapter 9.

5.7.2 Operation With a Device Different From a Teletypewriter

The Serial I/O Port, although intended for use with Teletypewriter, is certainly not limited for use with just that device. The user can conveniently connect any low speed bit serial data source to the Port. Plug replaceable daughter boards are available which allow the user

to fabricate his own Serial Port Adapter for converting his device signal levels to standard DTL/TTL levels used by the Processor. The built-in One Millisecond Clock may be adjusted ±10% for special applications. If the user's device supplies a clock, this may be conveniently connected into the Model 1 Interrupt System either as a replacement for the built in clock or in addition to the clock.

CHAPTER 6
MEMORY SYSTEM

6.1 INTRODUCTION

The Model 1 Memory System, Figure 6-1, features highly reliable 2048 by eight-bit byte, 1.0 microsecond memory modules. The system is modularly expandable to 16,384 bytes of storage (eight modules). Two types of plug compatible memory modules are available; Core Memory Modules and Read-Only-Memory (ROM) Modules.

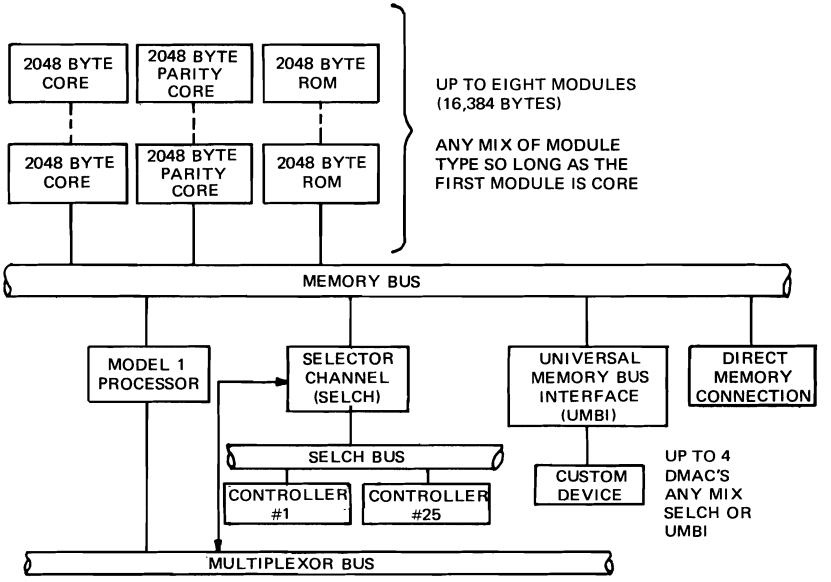


Figure 6-1. Model 1 Memory System

Core Memory Modules use field proven 3D, three-wire technology. The memory plane uses 23 mil ferrite cores and is mounted, complete with access and readout circuits, on a single standard INTERDATA circuit board. A 2048 by nine bit Parity Core Memory Module is available.

The Model 1 Read-Only-Memory (ROM) is plug compatible with Core Memory Modules. It is a non-volatile, hardwired, braided transformer memory having access characteristics identical to the core modules.

The first module in a Model 1 System must be a Core Memory. The remaining seven modules in a fully expanded system may be any combination of Core and ROM Modules. Similarly, Parity Core Modules may be mixed with non-Parity Modules in any fashion. (Parity ROM Modules are not available).

Up to four Direct Memory Access Channels (DMAC) can be added to a Model 1 Memory System. The DMACs operate over the common Memory Bus on a cycle stealing basis through a Direct Memory Access Port which is built into the Processor. Data rates through a DMAC can achieve 1,000,000 bytes per second.

Two types of Direct Memory Access Channels are available from INTERDATA. The Selector Channel (SELCH) operates with any standard INTERDATA peripheral device controller directly to/from the Memory System. Up to 25 controllers can connect to a single SELCH. The program sets up the SELCH by sending it a starting address, a byte count, and a GO Command. The SELCH coordinates the data transfer, autonomously, on a cycle stealing basis, and interrupts the Processor when the transfer terminates. Detailed description of the SELCH is covered in Chapter 5.

The second DMAC offered is the Universal Memory Bus Interface (UMBI). This is a general purpose interface to the Memory Bus which facilitates custom DMAC design. The UMBI has all necessary Memory Bus drivers and receivers, all Memory Bus control logic and a buffer Data Register. The remaining portion of the circuit board has IC sockets which allow the user to conveniently implement his custom logic. A detailed description of the UMBI is covered in Chapter 5.

The user may, if he wishes, connect directly to the Memory Bus using his own interface.

6.2 CORE MEMORY MODULES

INTERDATA Model 1 Core Memory Modules are three-wire, 3D, ferrite core memories which use 23 mil cores. Each Core Memory Module contains 2048 bytes of storage. Parity Core Memory Modules have a ninth bit plane in the stack corresponding to the Parity Bit. There are 64 X-axis wires and 32 Y-axis wires common to all bit planes. Each bit plane has its own sense/inhibit wire. In a Write operation, inhibit current is passed over this wire to write a ZERO (inhibit current is not passed over this wire to write a ONE). In a Read operation, the bit readout is sensed over the sense/inhibit wire. The block diagram in Figure 6-2 illustrates a Core Memory Module.

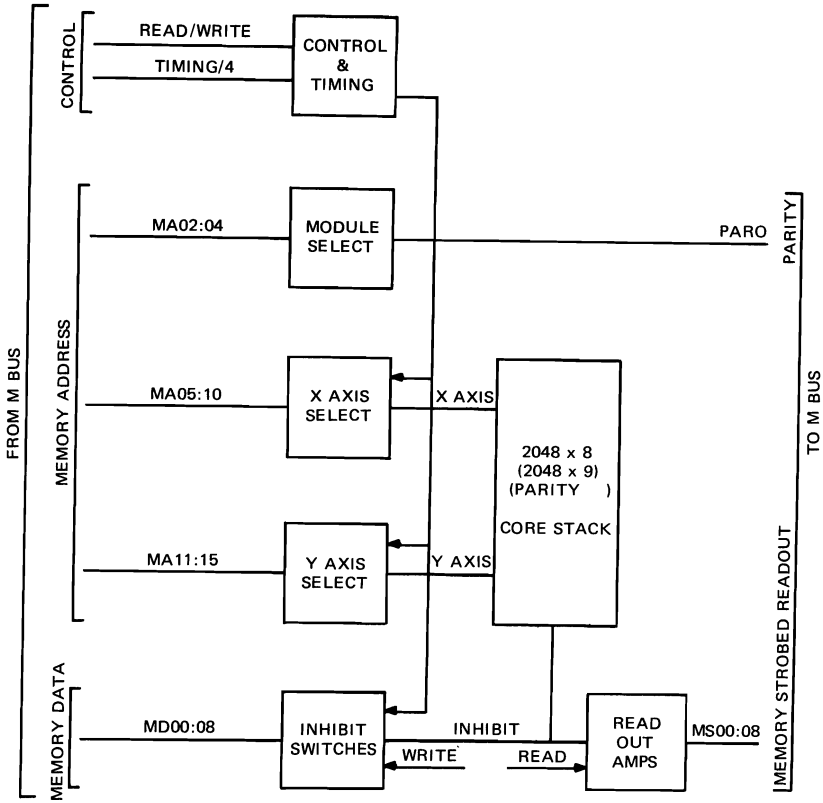


Figure 6-2. Core Memory Module

6.3 READ–ONLY–MEMORY MODULE

The Model 1 Read-Only-Memory Module is illustrated in Figure 6-3. The ROM is a plug replacement for a Model 1 Core Memory Module and is useful for guaranteed non-volatile program storage. The ROM uses a braided transformer array which is hardwired at time of manufacture from data supplied by the customer. The ROM has access characteristics identical to Model 1 Core Memory Modules. Parity ROM Modules are not available.

6.4 PARITY OPTION

The parity control logic is built into the Model 1 Processor. The parity logic is enabled/disabled by the currently addressed Memory Module via a Parity Enable (PAR0) line in the Memory Bus. The Processor's parity circuit computes or checks parity whether the current Memory operation is from some Direct Memory Access Channel or the Processor. The Processor does not check parity on a Read operation unless the addressed Memory Module has the optional Parity feature. This allows both Parity and non-Parity

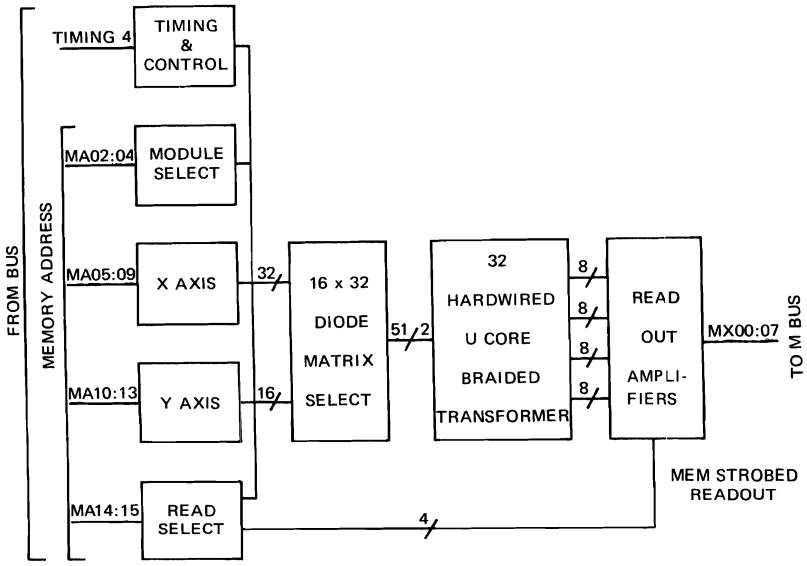


Figure 6-3. ROM Module 2048 By Eight-Bit

Memory Modules to be mixed in a given Memory System. The output of the parity check circuit is connected to Interrupt Line 3 in the Processor when a Parity Memory Module is installed in the system. This interrupt line, as are all others, is individually maskable. If a parity failure occurs and if the parity interrupt is enabled, the Processor stores the Location Count, the Carry Bit, and the Enable Bit at address X'110' and X'111' and begins the interrupt service program at address X'112'.

6.5 MEMORY BUS

The Memory Bus provides the communication paths between the Memory Modules, the Processor, and the Direct Memory Access Channels. The bus totals 42 lines. See Figure 6-4.

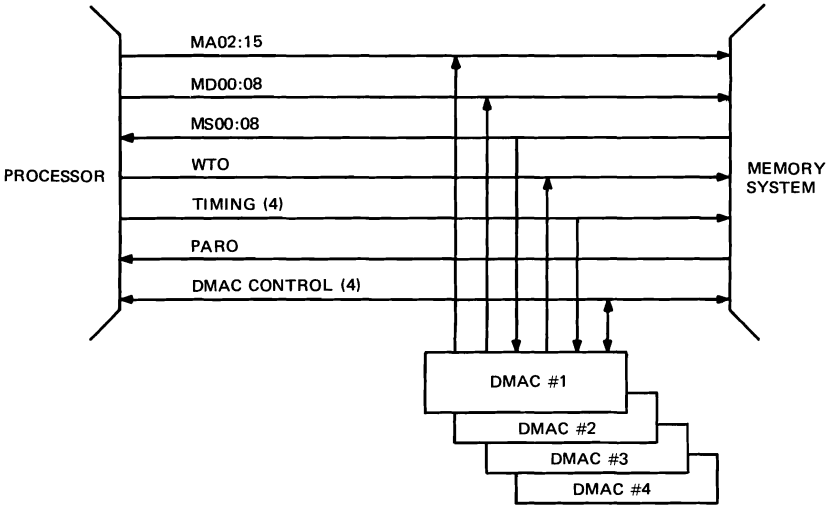
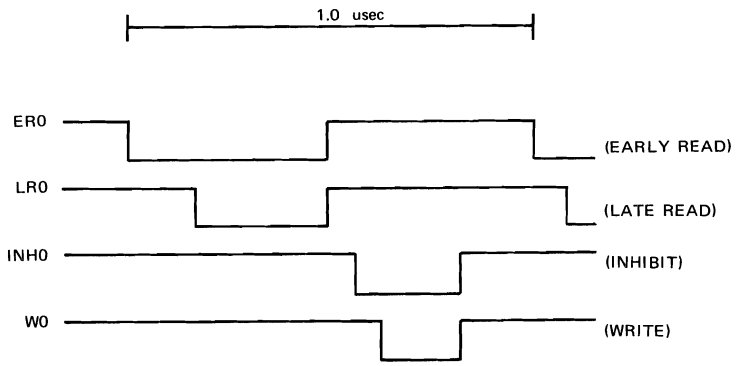


Figure 6-4. Model 1 Memory System Diagram

The four DMAC Control Lines are used for establishing whether the Processor or one of the DMA Channels is selected for the next memory cycle. Memory service is granted on a priority basis. The Processor is always last in priority. The four DMA Channels are assigned priority on a parallel daisy-chain basis where the device "closest" to the Processor is highest in priority.

The Parity Enable (PAR0) Line enables the parity control logic whenever a Parity Memory Module is addressed. The parity logic is disabled when a Memory Module without the Parity option is addressed.

The four Timing Lines required by the Memory Modules are generated by the Processor whether it is selected for the current memory cycle or not. Memory timing is identical for ROM or Core Memory Modules. The Read access characteristics of each are identical. The total memory cycle time is 1.0 microsecond. The memory access time is 0.5 microseconds.



The Write (WTO) Line defines whether the current memory cycle is a Read or a Write cycle.

The Memory Strobed Readout lines (MS00:08) carry the strobed readout from the addressed memory location on Read operations. These lines are idle on Write operations (MS08 corresponds to the Parity Bit).

The Memory Address Lines (MA02:15) carry a fourteen-bit address which identifies a unique memory location of the 16,384 locations in a fully expanded system.

Only one device may communicate with the Memory Bus at a time. When a DMA Channel requests memory, the Processor stops all processing and disconnects itself from the Bus. At the same time, the Processor sends an Enable signal which allows the highest priority DMA Channel to connect itself to the Bus. The Processor generates all Memory Timing signals and generates or checks Memory Parity if a Memory Parity Module is addressed. The DMA Channel must provide Memory Address (MA02:15), Memory Data (MA00:07), and the Read/Write Control Line (WTO).

6.5.1 Priority On The Memory Bus

The Processor is assigned lowest priority on the Memory Bus. It will always stop and give the next available memory cycle to the DMA Channels whenever anyone of them requests service. Priority among the four DMA Channels is established on a parallel daisy-chain basis. See Figure 6-5.

Any DMA Channel may request memory service at any time. The Processor scans the REQ0 line during every memory cycle. If it is active, the Processor responds by generating the Enable Line (ENO). The ENO line is applied simultaneously to all DMA Channels. The rising edge of EN0 marks the time the selected DMA Channel may switch itself onto the Memory Bus. See Figure 6-5.

E. M. GETTEL

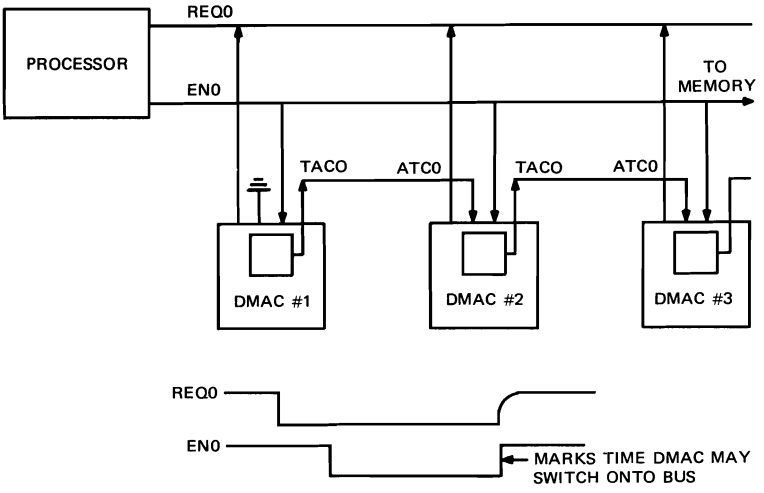


Figure 6-5. Example of Memory Bus Priorities

The EN0 signal is also passed daisy-chain fashion (ATC0/TAC0) from the top priority device, to the next highest priority device, and so on, until it is captured by a DMA Channel that requested service. The captured daisy-chain pulse ANDed with the rising edge of EN0 selects the DMA Channel. At this time, the selected DMA Channel must be prepared to execute the Memory operation.

6.5.2 Interfacing To The Memory Bus

Customers may interface their circuits to the Memory Bus using the Universal Memory Bus Interface. (See Chapter 5.) However, those who wish to, may interface directly according to the rules in this section.

All signals on the Memory Bus are DTL/TTL level compatible. A logical ONE or a true condition corresponds to a low level, V = 0.4. A logical ZERO or a false condition corresponds to a high level, V = 2.5 volts. The customer interface is permitted two DTL/TTL standard loads or two DTL open collector power gate OR ties onto lines on the Memory Bus. The customer may not OR tie onto any of the four Timing Lines, the Parity Enable (PAR0) line, or the Memory Strobed Readout Lines (MS00:08). Pull-up resistors are supplied for all lines by the Processor.

Clay Archer

NOTE

The INTERDATA Memory Bus may not be physically extended beyond the Standard Configuration limits described in Chapter 9.

Reference is made to Figure 6-6 covering detailed memory timing on all lines in the Memory Bus.

Refer to Figure 6-6, Model 1 Memory Bus Timing, and to Figure 6-4, Model 1 Memory System diagram. Figure 6-6 illustrates the activity on the Memory Bus for both Read and Write operations in detail. Back Panel pin assignments for an expansion slot in a standard INTERDATA chassis are shown in Figure 6-6. Note all connections related to the Memory Bus are in Column Zero. An expansion slot will accommodate either I/O or Memory.

REQUEST (REQ0)

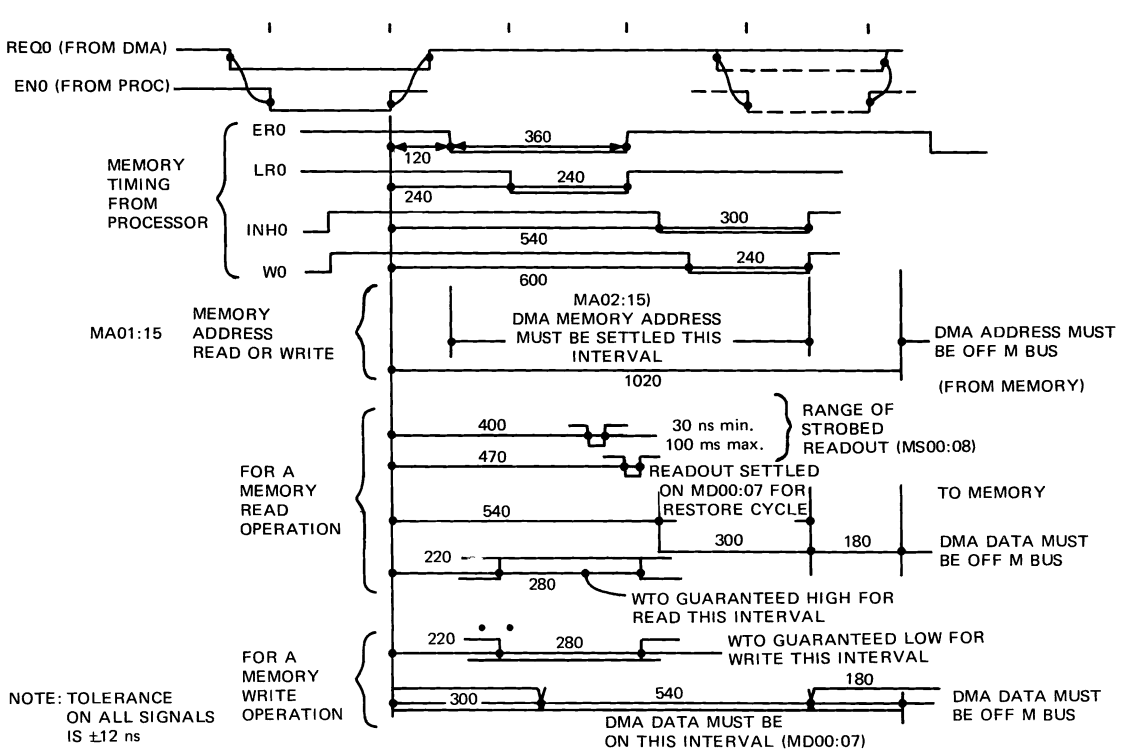
Up to four DMACs OR tie onto the REQ0 line. Any DMAC may activate REQ0 at any time. The system will not tolerate temporary false outputs on REQ0. The REQ0 line must be released within 500 nanoseconds following the rising edge of EN0 unless the DMAC wants two consecutive memory cycles.

ENABLE (EN0)

The Processor responds to a REQ0 by activating EN0. The delay between REQ0 and EN0 will be in the range 30 nanoseconds to 960 nanoseconds depending on when in the current memory cycle REQ0 is activated. The maximum period of EN0 is 960 nanoseconds (when REQ0 is steadily active). The width of EN0 is 240 nanoseconds.

EN0 is used to generate the daisy-chain priority loop through all DMACs in the system. The daisy-chain loop begins at the highest priority DMAC and propagates to the lower priority DMACs until it is "captured" by some DMA which requested service.

The "captured" daisy-chain pulse is ANDed with the rising edge of EN0 to set the Select flip-flop in the DMAC. Figure 6-7 illustrates a suitable circuit for the daisy-chain select request circuits.



FIELD 1

P16	P5	GND
N16	A2	B2
WT0		
MD000	C2	D2
010		
020	E2	F2
030		
040	G2	H2
050		
060	I2	J2
070		
080	K2	L2
MS080		
070	M2	N2
060		
050	O2	P2
040		
030	Q2	R2
020		
010	S2	T2
000		
TEMPB		V2
TEMPA	P5	GND

FIELD 0

REQ0	P5	GND
EN0	EXB2	EXD2
ACT0		
TAC0	EXA2	EXC2
MA150		
140	SCLRO	SPRC
130	SPRA	SPRB
120	SYNO	ATNO
110	RACKO	TACKO
100	SRO	CL070
090	DRO	060
080	DAO	
070	ADRSO	CMDO
060	DRL030	DRL070
050	020	060
040	010	050
030	000	040
020	DAL030	DAL070
PAR0	020	060
LR0	010	050
ER0	000	040
INH0	P15	N15
W0	P5	GND

Figure 6-6. Model 1 Memory Bus Timing

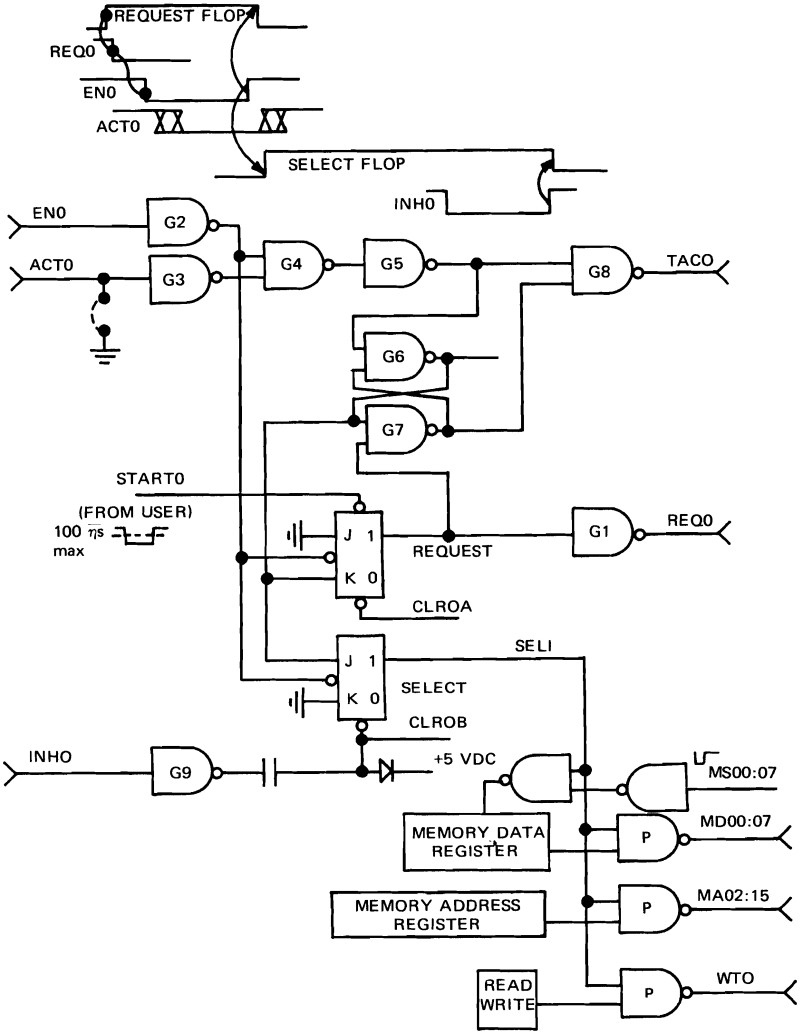


Figure 6-7 Daisy-Chain Select Request Circuits

The selected DMACs memory cycle begins on the rising edge of EN0 and it terminates on the rising edge of INH0 (described later).

MEMORY TIMING LINES

Early Read (ER0) Late Read (LR0)	These two lines control the Read Current Switching in the addressed Memory Module.

INHIBIT (INH0) WRITE (WT0)	These two lines control the Write Current Switching in the address Memory Module. Memory Bus timing is identical whether the addressed Memory Module is a core memory or a ROM. The Memory Timing Lines are available to the Memory Bus Interface. The rising edge of INH0 is particularly useful since it marks the end of the effective memory cycle. The DMAC may disconnect itself from the Memory Bus at this time.
MEMORY ADDRESS LINES (MA02:15)	The selected DMAC may activate the MA lines anytime following the rising edge of EN0. However, it must guarantee the MA lines settled over the interval beginning 120 nanoseconds after EN0 to the rising edge of INH0. Further, the DMAC must release the MA lines within 180 nanoseconds after the rising edge of INH0.
WRITE (WT0)	This line must be settled during the interval beginning 220 nanoseconds after the rising edge of EN0 to 500 nanoseconds after the same edge. If WT0 is active (low), a Write operation will be performed.
MEMORY STROBED READOUT LINES (MS00:08)	The MS Lines carry the pulsed single-rail Memory Readout on Read operations. The MS lines are inactive on Write operations. The pulsed readout varies in width from 30 nanoseconds to 100 nanoseconds and in position from 400 nanoseconds to 470 nanoseconds following the rising edge of EN0. The selected DMAC is required to register the Memory Readout and return the data on the MD Lines for the restore position of the cycle.
MEMORY DATA LINES (MD00:08)	The MD Lines carry data to be written to the addressed Memory Module. They also carry data to be restored to the addressed Memory Module. If the

operation is Write, the MD Lines must be settled for the interval beginning 300 nanoseconds following EN0 to the rising edge of INH0. If the operation is Read, the strobed readout from the MS Lines must be returned to the MD Lines by the falling edge of INH0 and must remain so until INH0 rises. Regardless of the operation (Read or Write), the DMAC must release the MD Lines within 180 nanoseconds after the rising edge of INH0.

Note that MD08 corresponds to the Parity Bit. The Processor always controls this line. The DMAC should not OR tie onto MD08.

6.5.3 Memory Bus Timing

The Request flip-flop is set by START0 when the user's circuits require memory service. (START0 should be at least 30 nanoseconds wide and no wider than 100 nanoseonds). The REQ0 line is driven active through Gate G1 when the Request flip-flop is set.

Gates G2 through G7 form the daisy-chain capture circuit which establishes DMAC priority. Gates G6 and G7 are a contention circuit which will capture the ACT0 pulse if the Request flip-flop is set or it passes ACT0 onto the next DMAC (as TAC0) through G8 if the Request flip-flop is not set. The contention circuit will ignore a change in the Request flip-flop if it changes during the ACT0 period.

The daisy-chain delay through one DMAC is 48 nanoseconds maximum, assuming TTL logic, 12 nanoseconds maximum delay per stage.

If the circuit captures the daisy chain pulse, the Request flip-flop is toggled reset on the rising edge of EN0 and the Select flip-flop is toggled set at the same time. The Select flip-flop is used to switch the DMAC onto the Memory Bus. The Select flip-flop is cleared on the rising edge of INH0 by Gate G9 which marks the end of the memory cycle.

The ACT0 Line is grounded on the top priority DMAC. On all other DMACs, the ACT0 input is driven by TAC0 from the next higher priority DMAC.

The two flip-flops, REQUEST and SELECT, should be initialized (CLR0A and CLR0B) to a reset condition on Power Up by the user's circuits. (The SCLR0 Line on 117-0 is useful for this function.)

CHAPTER 7
CONTROL PANEL

7.1 INTRODUCTION

This chapter discusses the Model 1 Control Panel, illustrated in Figure 7-1, which is supplied as part of the Model 1 Processor. The Control Panel provides the system operator with visual indications of the state of the Processor as well as a wide variety of manual control over the Model 1 System. It is a particularly useful tool for entering data into memory, for writing and debugging programs, and for performing routine system maintenance.

Although the Standard Control Panel is a powerful tool, it is not a mandatory part of a useful Model 1 System. Therefore, the Standard Control Panel is designed to be plug replaceable by a lower cost Auto-Control Panel when the day-to-day use of the system does not require the convenience of the Standard Control Panel. The Auto-Control Panel has a key operated ON-OFF-AUTO Switch. It has no capability to display registers or load/unload memory directly. The low-cost Auto-Control Panel may be plug replaced with a Standard Control Panel if program modification and system maintenance is required. This chapter discusses the Standard Control Panel, the Auto-Control Panel, and operating procedures related to them.

7.2 STANDARD CONTROL PANEL DESCRIPTION

The Standard Control Panel, Figure 7-1, is a RETMA standard 5-1/4" x 19" panel which is plug removable from a Model 1 Processor. It provides all necessary manual control over a Model 1 System. The Standard Control Panel includes the following set of control elements:

Twenty-Six (26) Indicator Lamps

Fourteen (14) Data Switches

Six (6) Control Switches

Key Operated Security Lock

A functional description of each of these is given in this section.

7.2.1 Indicator Lamps

The 26 indicator lamps on the Standard Control Panel continually display the state of the following registers or conditions:

Accumulator	A00:07	8 bits
Location Counter	LOC02:15	14 bits
Interrupt Enable	ENABLE	1 bit

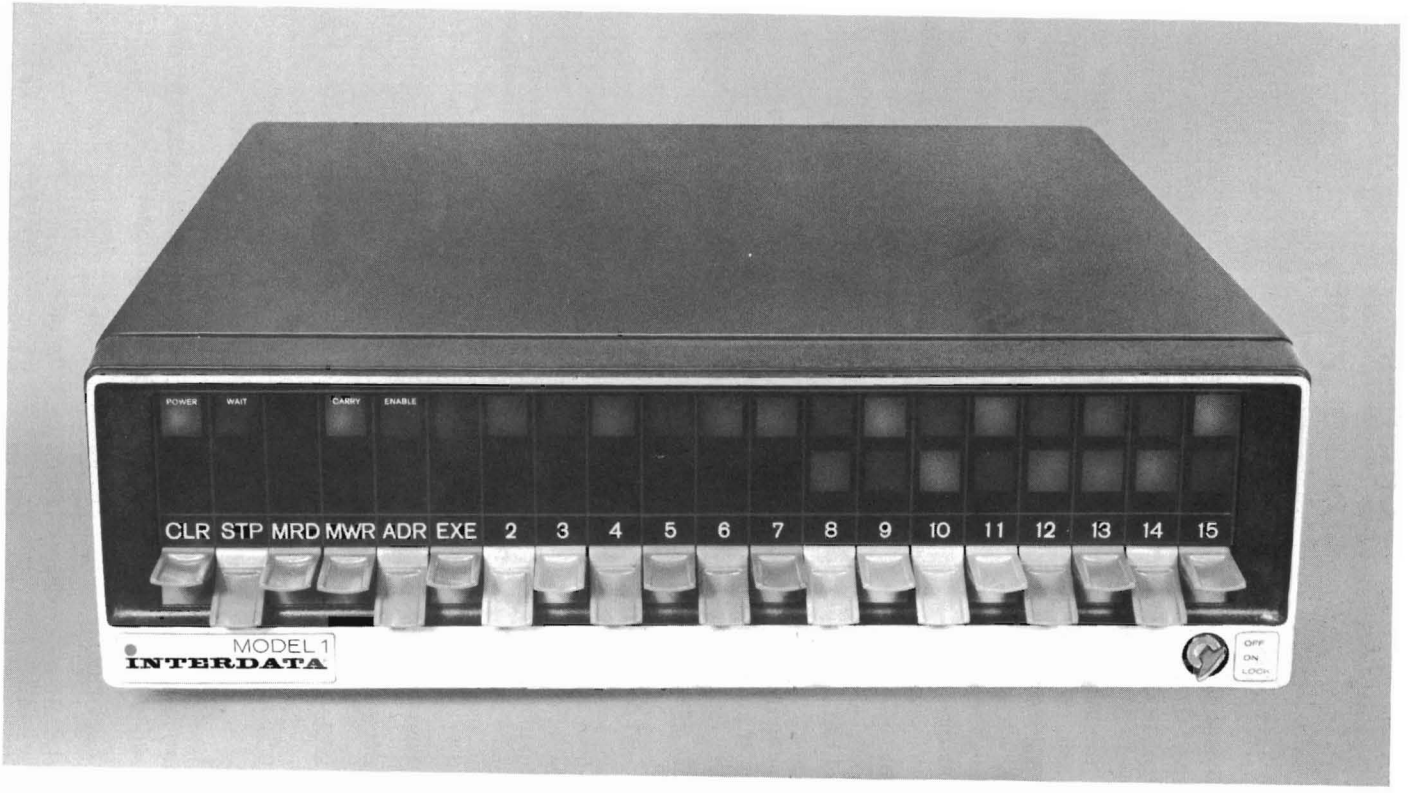


Figure 7-1. Model 1 Control Panel

Carry Bit	CARRY	1 bit
Wait Mode	WAIT	1 bit
Power On	PWR	1 bit

7.2.2 Data Switches

The Fourteen latching Data Switches (02:15) are used to enter a memory address to the Model 1 System. The least significant eight Data Switches (08:15) are used to load an eight-bit data byte to memory. The state of these switches may be loaded to the Accumulator at program command.

7.2.3 Control Switches

The six Control Switches on the Standard Control Panel are active when the key operated security lock is in the ON position only.

CLEAR (CLR)	The momentary CLR Switch causes the Model 1 System to be initialized. After the initialize operation, all device controllers on the System Multiplexor Bus are cleared, the Carry Bit and the Interrupt Enable Bit are cleared, the Interrupt Queue Register is cleared, the instruction at location zero will be executed when the Processor restarts.
EXECUTE (EXE)	The momentary EXE Switch causes the Processor to perform the Control Panel operation indicated by all other Control Switches except CLR. The other switches have no effect unless EXECUTE is depressed.
STOP (STP)	The latching STP Switch causes the Processor to stop program execution.
ADDRESS (ADR))	The latching ADR Switch causes the state of Data Switches 02:15 to be loaded to the Processor's Location Counter which is displayed on indicator lamps.
MEMORY WRITE (MWR)	The latching MWR Switch causes the state of Data Switches 08:15 to be written into memory at the address contained in the Location Counter. After the Write operation is performed, the Location Counter is incremented by one. The eight-bit data byte written is in the Accumulator.

MEMORY READ (MRD) An eight-bit data byte is read from the memory address contained in the Location Counter and is loaded to the Accumulator. After the Read operation, the Location Counter is incremented by one.

NOTE

Only one of the three Switches ADR, MWR, or MRD should be active at any given time.

7.2.4 Key Operated Security Lock

This is a three-position key-operated locking switch which switches primary power on and off the Model 1 System and which can disable all Control Switches on the panel to eliminate any accidental manual input to the system.

OFF The primary power to the Model 1 System is turned off.

ON The primary power to the Model 1 System is turned on and all Control Switches on the panel are enabled.

LOCK The primary power to the Model 1 System remains on but all Control Switches to the panel are disabled.

7.3 CONTROL PANEL OPERATING PROCEDURES

1. To bring power up on the system,
   a. Turn the key-operated security lock clockwise from the OFF position to the ON position.
   b. The Processor will unconditionally execute the instruction at location zero.
2. To shut power down to the system,
   a. Turn the key-operated security lock clockwise to the OFF position.
   b. This removes AC power from the system and forces a Primary Power Fail Interrupt to the Processor.
   c. The program is required to store A in memory and to issue a Command Power Down (POW) within five milliseconds of the interrupt if it is desired to save the current machine status.

NOTE

The hardware will eventually respond to a power down condition, even if the program does not. However, the content of A, C, E, and one memory location may be lost without adequate program support.

NOTE

Further details regarding program support for Control Panel are found in Chapter 9, covering Basic Model 1 programming.

3. To manually stop execution of a program.

   a. Depress the Stop (STP) Switch.

   b. Depress the momentary EXECUTE (EXE) Switch once. (The Processor stops executing the program, the Location Counter is pointing to the next instruction in sequence).

4. To manually start execution of a program: (The system must be in the STOP Mode)

   a. Enter the starting address of the program on the fourteen Data Switches.

   b. Depress the ADDRESS Switch.

   c. Depress the momentary EXECUTE (EXE) Switch. (The starting address of the program is now displayed in the Location Counter.)

   d. Release the STP Switch.

   e. Release the ADR Switch.

   f. Depress the momentary EXE Switch. (The Processor begins executing the program at the address entered in Steps 1 through 3.)

5. To manually execute a program, one instruction at a time:

   a. Stop the program as in Step 3 above, leaving the Stop (STP) Switch depressed.

   b. Insure that neither ADR, MWR, or MRD is depressed.

   c. Depress EXECUTE once for each instruction to be executed. After each instruction is executed, the Location Counter will point to the next instruction to be executed.

NOTE

When the system is in this mode, it will respond to external interrupts by storing the Location Counter, Carry, and Enable Bits in the fixed Memory Interrupt Service Table on any EXECUTE. The next EXECUTE will cause the Processor to execute the first instruction in the Service Table.

6. To read memory and display the results:

   a. The system must be in the STOP Mode.

   b. Enter the Memory Address to be read on the twelve Data Switches.

   c. Depress the latching ADDRESS Switch.

   d. Depress EXECUTE. (The address of the byte to be read is now displayed in the Location Counter.)

   e. Depress the MEMORY READ Switch.

   f. Depress EXECUTE. (The addressed memory location is read and the results displayed on the Accumulator. The Location Count is incremented once and points to the next address to be read. The operator may read consecutive memory locations by depressing EXECUTE once for each address.)

7. To write to memory from the Data Switches:

   a. The system must be in the STOP Mode.

   b. Enter the Memory Address to be written to on the fourteen Data Switches.

   c. Depress the ADDRESS Switch.

   d. Depress EXECUTE. (The Memory Address to be written to is now displayed on the Location Counter.)

   e. Enter the eight-bit Data Byte on the Data Switches 08:15.

   f. Depress the MEMORY WRITE Switch.

   g. Depress EXECUTE once. (The Data Byte written to the memory is displayed in the Accumulator, the Location Counter is incremented by one, and points to the next memory address to be written. The operator may write consecutive memory locations by changing the Data Switches as required and depressing EXECUTE once for each byte.)

8. To restart from the WAIT Mode:

   a. Depress the STOP Switch then the EXECUTE Switch.

   b. Release the STOP Switch.

   c. Depress the EXECUTE Switch. The program now begins running at the instruction following the WAIT Command.

7.4 CONTROL PANEL PROGRAMMING

The Control Panel's Data Switches 08:15 may be read by program control using the

COMMAND READ DATA SWITCHES

instruction. The programmer is cautioned that indeterminate results could occur if the Data Switches are changed at the same time the program executes this instruction due to contact bounce. This can be overcome by testing the switch several times over, for example, one second intervals.

7.5 AUTO–CONTROL PANEL

The Auto-Control Panel is a low-cost plug replacement for the Standard Control Panel. It has only a key-operated ON/OFF/AUTO Switch.

Systems with an Auto-Control Panel have a built-in Read-Only-Memory program designed to load data to the system from a teletypewriter through the Serial I/O Port. This is done when the key-operated Switch is turned from OFF to AUTO. The tape will begin loading at address X'100'. The tape must contain exactly 256 bytes. After the tape is loaded, the Processor begins executing instructions at address X'100'.

When power is applied to the system normally, by switching the key-operated switch from OFF to ON, the Processor will begin executing instructions at address X'017C'.

Refer to Chapter 10, Basic Model 1 Programming, for details of operating Auto-Control Panel Systems.

CHAPTER 8
PERIPHERAL DEVICES

8.1 INTRODUCTION

The Model 1 Processor's Multiplexor Bus (I/O Bus) is fully hardware plug compatible with all other INTERDATA Processors. Therefore, the full line of field proven peripheral device controllers and system modules are available for use with Model 1 Systems.

This Chapter lists all standard INTERDATA peripheral device controllers and system modules available for the Model 1. Periodically, significant new products are added to the INTERDATA Product Line which may not appear in this manual. Please check with any INTERDATA Sales Office for the latest additions.

Detailed programming and device specifications for Teletypewriters, High Speed Paper Tape Readers and Punches, and Card Readers are provided later in this chapter. Detailed programming and device specification on other INTERDATA products are available through the local Sales Offices on request.

8.2 PERIPHERAL DEVICES

The following peripheral devices are presently available. All peripheral devices are complete with the necessary interface to the Multiplexor Bus (I/O Bus).

8.2.1 Teletypewriters

Product Number	Description
1-420	ASR Model 33
1-422	ASR Model 35
1-424	KSR Model 33
1-425	KSR Model 35

8.2.2 Paper Tape and Card Equipment

Product Number	Description
1-410	Paper Tape Reader Unidirectional 300 cps, Rack Mounting with fanfold bins.

Product Number	Description
1-411	Paper Tape Reader Bidirectional, 300 cps, Rack Mounting with fanfold bins
1-415	Paper Tape Handler with 8" NAB Reels
1-412	Paper Tape Punch, 60 cps
1-413	Combination Reader/Punch
1-510	Card Reader, 200 cpm
1-550	Line Printer, 300 lpm 132 columns, 64 characters
1-445	X-Y Incremental Plotter: Plots 18,000 steps per minute, .01" or .005" sets 12" wide paper
1-442	X-Y Oscilloscope Display Plots point to point, 10 bits/axis resolution

8.2.3 Magnetic Storage Systems

Product Number	Description
1-660	Intertape System includes: Byte buffered controller with two cassette tape decks and power supply; transfer rate 300 cps; 250K bytes of storage per cassette. Mini Disc System includes: Power supply, cabling and mounting hardware. Transfer rate 60K bytes per second; average access time is 8.5 milliseconds; capacity 51K bytes per disc; 1 x 2 controller
1-740	Interface plus first disc drive
1-741	Second Disc Drive

8.2.4 System Modules

Product Number	Description
1-834	General Purpose Interface Module Basic input/output interface without buffering.
1-811	Byte Input/Output Module 8 stored data output lines, 8 data input lines, 8 control lines, 8 sense lines, 1 priority interrupt line.
1-812	Halfword Input/Output Module 16 stored data output lines, 16 data input lines, 8 control lines, 8 sense lines, 1 priority interrupt line.
1-820	Control Line Module 16 buffered control lines for activation of up to 16 external devices.
1-821	Relay Closure Module 16 isolated reed relay closures with storage registers for operation of medium power devices.
1-822	Decimal Indicator Module 4 Decades of decimal indicators.
1-823	Decimal Indicator Module 8 decades of decimal indicators.
1-824	Decimal Indicator Module 4 decades of decimal indicators with decimal point.
1-825	Decimal Indicator Module 8 decades of decimal indicators with decimal point.
1-830	Sense Line Module 16 sense lines which may be used for detecting the status of devices under external command.

Product Number	Description
1-831	Sense Switch Module 16 switches for manual entry of program variables.
1-832	Manual Data Entry Module 8 decades of thumb-wheel switches and a 'Data Entry' push button.
1-833	Manual Data Entry Module 16 decades of thumb-wheel switches and a 'Data Entry' push button.

8.2.5 Digital Input/Output Multiplexor Equipment

Product Number	Description
1-860	Digital Multiplexor Controller plus first expansion chassis (1-864). This is a universal interface for both input and output modules. One controller can support up to 2048 input lines and 2048 output lines.
1-861	128 Line Input Module Provides transformer isolated current sensing inputs from 128 points.
1-862	128 Line Output Module Provides open collector output and can drive 4 each 1-865 or user circuits.
1-863	128 Line Convenience Panel Provides 128 twisted pair, 24 guage wire wrap connection pins. Allows for additional line filtering and signal conditioning elements. Cabling from input and output modules to convenience panel is supplied.
1-865	Relay Module Provides 32 high power relay contacts (non-latching 5 Amps at 240VAC) for heavy duty switching.

Product Number	Description
1-864	Digital Multiplexor Expansion Chassis Provides slots for four 128-line input/output modules.

8.2.6 Data Communications Equipment, Character Buffered Half Duplex Line Adapters

Product Number	Description
1-022	Bell 801 Automatic Dialer Control for 4 lines (Group of 4 only).
1-023	Bell 103 Data Set Adapter (Requires 10-052)
1-024	Bell 201 Data Set Adapter (Requires 10-052)
1-025	Bell 202 Data Set Adapter (Requires 10-052)
1-026	Bell 301 Data Set Adapter (Requires 10-055)
1-027	Parity Option for 1-023/24/25/26
10-052	25' Cable between Connector Panel and data set.
10-055	25' Cable between Connector Panel and 301 data set.

8.2.7 Conversion Equipment

Product Number	Description
	Multichannel Analog to Digital Converters Provides fully interfaced conversion of bipolar analog signals, and can multiplex up to 64 analog channels in both random or sequential modes.

Product Number	Description
1-751	A-D Converter with Buffering 8 bit Resolution
1-752	A-D Converter with Buffering 10 bit Resolution
1-753	A-D Converter with Buffering 12 bit Resolution
1-754	Sample and Hold Amplifier (Maximum of 1 per 64 channels)
1-771	4 Channel Multiplexor (Requires 1-751 or 1-752 or 1-753)
1-772	8 Channel Multiplexor (Requires 1-751 or 1-752 or 1-753)
1-773	12 Channel Multiplexor (Requires 1-751 or 1-752 or 1-753)
1-774	16 Channel Multiplexor (Requires 1-751 or 1-752 or 1-753) Maximum of 4 Multiplexor Boards per System.
	Multichannel Digital to Analog Converters Provides fully interfaced conversion of digital data to bipolar analog signals. The system can address up to 16 channels in random or sequential modes. Each channel employs dual rank registers and an operational amplifier.
1-764	D-A Controller for up to 16 Channels. (Requires 1-761 or 1-762 or 1-763)
1-761	Dual Channel D-A Converter 8 bit Resolution

Product Number	Description
1-762	Dual Channel D-A Converter 10 bit Resolution
1-763	Dual Channel D-A Converter 12 bit Resolution

8.3 TELETYPE (WITH TELETYPE CONTROLLER) OPERATION AND PROGRAMMING

8.3.1 Device/Controller Description

Device

Model Numbers - ASR33 and ASR35

Data Rate - 10 Characters per second

Printer Width - 72 Characters maximum

Character Set - see Table 8-1

Paper Feed - Pin feed

Dimensions - W 22", D 18-1/2", H 8-318" (without Stand) overall ASR33

W 38-1/2", D 24", H 38-1/2" (includes Stand) overall ASR35

Weight - 44 pounds desk top ASR33

225 pounds (includes stand) ASR35

Power Requirement - 115VAC 60 Hz

9A start up
2A running } - ASR35

12A start up
4A running } - ASR35

115VAC 50 Hz models are available

Controller

INTERDATA Part Number 32-120F01

Size - Single 10 inch INTERDATA standard printed circuit board

Back Panel Slots - requires one standard expansion slot

Power Requirement - 1A + 5VDC
.75A + 16VDC

Weight - 1-1/2 pounds (approx.)

Programming examples do not refer to a Teletype connected to the Serial I/O Port although the functional characteristics of the Teletype are the same. For Serial I/O Port programming examples, see Section 10.4.6.

TABLE 8-1
TELETYPE/ASCII/HEX CONVERSION TABLE

HEX (MSD)					8	9	A	B	C	D	E	F
(LSD)	Teletype Tape Channels →			8	Depends upon parity							
				7	0	0	0	0	1	1	1	1
				6	0	0	1	1	0	0	1	1
				5	0	1	0	1	0	1	0	1
	4	3	2	1								
ø	0	0	0	0	NULL	DC_0	SPACE	0	@	P		
1	0	0	0	1	SUM	X-ON	!	1	A	Q		
2	0	0	1	0	EOA	TAPE ON	"	2	B	R		
3	0	0	1	1	EOM	X-OFF	#	3	C	S		
4	0	1	0	0	EOT	TAPE OFF	$	4	D	T		
5	0	1	0	1	WRU	ERR	%	5	E	U		
6	0	1	1	0	RU	SYNC	&	6	F	V		
7	0	1	1	1	BELL	LEM	'	7	G	W		
8	1	0	0	0	FE_0	S_0	(	8	H	X		
9	1	0	0	1	HT/SK	S_1	)	9	I	Y		
A	1	0	1	0	LF	S_2	*	:	J	Z		
B	1	0	1	1	VT	S_3	+	;	K	[		
C	1	1	0	0	FF	S_4	,	<	L	\		ACK
D	1	1	0	1	CR	S_5	-	=	M	]		ALT. MODE
E	1	1	1	0	SO	S_6	.	>	N	↑		ESC
F	1	1	1	1	SI	S_7	/	?	O	←		DEL

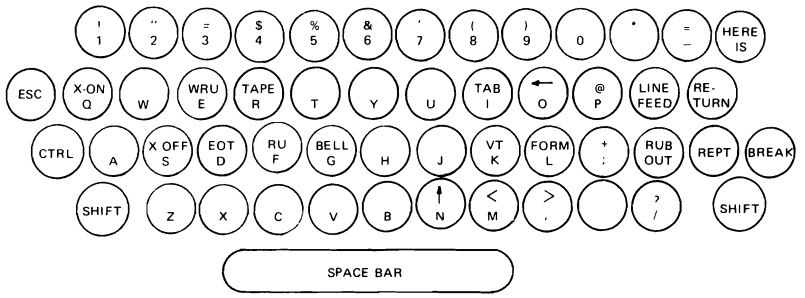


Figure 8-1. Teletype Keyboard Layout

8.3.2 Power Control

A three position power switch is located to the right and below the keyboard. When rotated left to the position marked LINE, power is applied to the Teletype and the device is logically connected to the Processor. When rotated to the right to the position marked LOCAL, the unit is powered, but disconnected from the Processor.

8.3.3 Status and Commands

Table 8-2 illustrates the Teletype Status and Command Byte coding.

A Sense Status Instruction (SS) is used to transfer the status byte from the device controller to the accumulator. In general, testing the status byte for all bits zero during READ operations is required.

8.3.4 Device Number

Teletypes supplied by INTERDATA are normally assigned Device Number 2. The device number can be changed, or additional Teletypes added, as needed by a minor change to the teletype controller. Refer to the Maintenance Manual for details.

TABLE 8-2.
TELETYPE STATUS AND COMMAND BYTE DATA
HEX ADDRESS 02

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE			BRK		BSY	EX		DU
COMMAND BYTE	DISABLE	ENABLE	UNBLOCK	BLOCK	WRT	READ	PWR ON	PWR OFF

BRK — The Break bit is set when the Break key on the Teletype is depressed, or the Teletype is logically disconnected from the Processor.

BSY — The significance of the Busy bit depends upon whether a Read or a Write operation is in progress. During Write mode, BSY is normally low, and goes high only while data is being received by the device. During Read mode, BSY is normally high, and goes low only when data has been received from the device, but not yet been transferred to the Processor. During Read mode, BSY goes high again as soon as the Processor accepts the data.

EX — The Examine bit is set whenever BRK is set.

DU — The Device Unavailable bit is set whenever the Teletype power is off, the Teletype is in LOCAL mode, or power is not connected to the Teletype.

DISABLE — This command disables the Device Interrupt to the Processor from the Device Controller.

ENABLE — This command enables the Device Interrupt to the Processor from the Device Controller.

UNBLOCK — This command enables the printer to print data entered via either the keyboard or the tape reader.

BLOCK — This command disables the feature described above.

WRT, READ — The Write and Read commands are used to define the significance of the BSY bit.

PWR ON, PWR OFF — The Power On and Power Off commands are significant only with those Teletypes provided with an optional Power Control Box. The option permits the Teletype power to be enabled or disabled under program control.

8.3.5 Interrupts

The interrupt associated with the Teletype is a data-ready, or ready-to-transfer interrupt. That is, when enabled, an interrupt is generated by the Teletype controller whenever it is ready to execute a data transfer with the Processor. In the WRITE mode, an interrupt occurs when the controller is ready for another character to send to the Teletype. In the READ Mode, an interrupt occurs when the controller has assembled a character for transfer to the Processor.

Note that when changing from the READ mode to the WRITE mode with interrupts enabled, an interrupt occurs as soon as the controller is ready to receive the first character for output.

8.3.6 Initialization

When the CLR button on the Processor Display Panel is depressed, the following occurs:

1. The DSBL, BLK, and READ commands functions are set.
2. The BSY status bit is set.
3. The BRK, EX, and DU status bits are reset.
4. Any pending interrupts are cleared.

8.3.7 ASR-35 Features

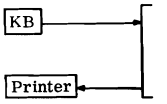
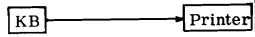
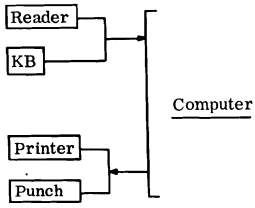
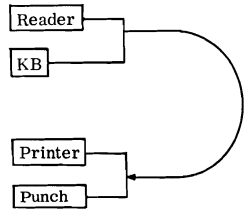
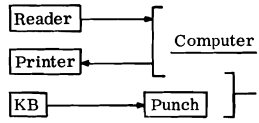
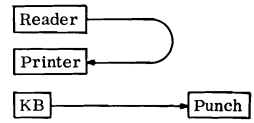
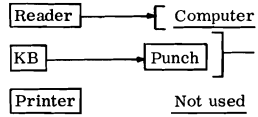
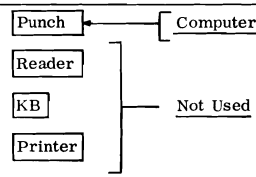
The ASR-35 is a ruggedized or heavy-duty version of the ASR-33. The programming of an ASR-35 is identical to an ASR-33. The operation of an ASR-35 is similar to an ASR-33 with the following exceptions:

1. The Tape Reader and Tape Punch controls are different as explained later in this description.
2. The ASR-35 has a mode control switch to the left of the keyboard. The meaning of the 5 positions of this switch is illustrated in Table 8-3.
3. Several additional keys, such as Local Line Feed, are provided. The meaning of these keys is self-explanatory.

8.3.8 Paper Tape Reader

The tape reader is controlled by a three-position switch on the reader. The three positions are START, STOP, and FREE. When the switch is moved to the START position, any tape in the reader is advanced continuously at a 10 character per second rate. Tape motion continues until the reader switch is moved to the STOP position or the tape runs out of the reader.

TABLE 8-3.
35 ASR OPERATING MODES

Mode	Line	Local
K (Keyboard)		
KT (Keyboard Tape)		
T (Tape)		
TTs (Tape to Tape Send) Non ASCII Tapes		Not Applicable
TTr (Tape to Tape Rev) Non ASCII Tapes		Not Applicable

In the STOP position, tape motion can be controlled by program, assuming the teletype power switch is in the LINE position. The specific control characters which affect the reader are X-ON (X'91') which starts the tape motion, and X-OFF (X'93') which stops tape motion.

The programmed steps required to start the reader are as follows:

```
LI      2       ADDRESS DEVICE
ADR
LI      '98'    SET WRITE MODE
OC
```

```
LI      '91'        GET X-ON CHAR
WDS
B       ERROR
LI      '94'        SET READ MODE
OC
RDS                 READ A CHAR
B       ERROR
```

The programmed steps required to stop the reader are as follows:

```
LI      '98'        SET WRITE MODE
OC
LI      '93'        GET X-OFF CHAR
WDS                 STOP TAPE
B       ERROR
```

Note that when stopping the tape reader under program control, the tape may advance 1 or 2 characters between the time the XOFF character is issued and the tape comes to a complete halt. Similarly on starting a tape, 1 or 2 characters may be missed before synchronization is attained. Therefore, tapes to be read under program control should be formatted to account for the start/stop characteristics of the reader.

The above procedures apply to both the ASR-33 and the ASR-35 teletypes. The ASR-35 tape reader is enabled, however, only when the ASR-35 Mode Switch is in the KT, T, or TTS positions. See Table 8-3 for details.

8.3.9 Paper Tape Punch

The ASR-33 tape punch can be manually turned on at any time by depressing the LOCK "ON" button on the punch unit. Once turned on in this fashion, all data output to the Teletype is unconditionally printed and punched. Note that non-printing characters transferred to the Teletype will be punched, but no image will be printed, and the printer carriage will not advance.

To manually turn the punch off on the ASR-33, the following steps are required:

1. Turn the power switch to LOCAL mode.
2. Depress the UNLOCK key on the tape punch.
3. Strike the $\overline{\text{TAPE}}$ key while the CTRL key is depressed.

If the ASR-33 is not in a LOCK"ON" mode (depress UNLOCK to release the LOCK "ON" mode), and the power switch is in the LINE position, the tape punch can be controlled via program.

The specific control characters which affect the punch are TAPE ON (X'92'), which starts the punch and TAPE OFF (X'94'), which stops the punch. The punch controls are achieved by outputting the appropriate character to the teletype. Note that the TAPE ON and TAPE OFF characters, themselves, will get punched on the tape.

The tape punch can be manually started in an alternate way. If the punch is not already on, strike the TAPE key with the CTRL key depressed, and the power switch in LOCAL mode. This technique is equivalent to transferring a TAPE ON character to the Teletype from the Processor.

The ASR-35 tape punch is enabled only when the ASR-35 Mode Switch is in the KT or TTR position. In these modes, the punch is controlled via TAPE and $\overline{\text{TAPE}}$ keys, and TAPE ON and TAPE OFF characters as described above. Refer to Table 8-3 for details. Following the program transfer of a TAPE ON character to start the ASR-35 TAPE punch, the program should output 2 or 3 rubout characters (X'FF') to achieve data synchronization prior to punching the data.

8.3.10 Data Formats

The format of data transferred to and from the teletype is as follows:

1. When reading data from a tape, each 8-bit tape character is transferred to the Processor as one 8-bit data byte.
2. When reading data from the keyboard, one data byte is transferred for each key depressed. The data is ASCII code for the particular character, in which the most significant bit for the character is a one or zero to achieve even parity for that character. In general, programs which read Teletype data mask the most significant bit to zero, and manipulate 7-bit ASCII codes in memory. See Table 8-1.
3. When transferring characters from the Processor to the Teletype Printer, the most significant bit of each character can be either one or zero since it has no effect on the Teletype.
4. When transferring characters from the Processor to the Teletype punch, each 8-bit data byte is punched as one tape character. The most significant bit is punched as specified in the data byte.

8.3.11 Program Examples

Typical routines for transferring data to and from the teletype are as follows:

EXAMPLE 1

```
*THIS IS A SUBROUTINE WHICH READS A BYTE FROM THE
*TELETYPE
*THE CALL IS BAL READ
*BYTE IS RETURNED IN THE ACCUMULATOR
*
READ    DC      '0000'
        LI      2           ADDRESS TTY
        ADR
        L       CMD         LOAD COMMAND
        OC                  SET READ MODE
        RDS                 READ
        B       ERROR       EXIT IF ERROR
        B       READ, I     EXIT NORMAL
CMD     DC      'A4'        DISABLE, UNBLOCK, READ
        END
```

EXAMPLE 2

```
*THIS IS A SUBROUTINE WHICH OUTPUTS A BYTE TO THE
*TELETYPE
*ENTER THE ROUTINE WITH THE BYTE IN THE
*ACCUMULATOR
*THE CALL IS BAL WRITE
WRITE   DC      '0000'
        ST      SAVE+1      SAVE DATA BYTE
        LI      2           ADDRESS TTY
        ADR
        L       CMD         LOAD COMMAND
        OC                  SET WRITE MODE
SAVE    LI      0           GET DATA BYTE
        WDS                 WRITE CHAR
```

```
            B       ERROR
            B       WRITE, I    EXIT

CMD         DC      '98'        DISABLE, BLOCK, WRITE
                                COMMAND
```

8.4 HIGH SPEED PAPER TAPE READER/PUNCH OPERATION AND PROGRAMMING

8.4.1 Introduction

This section provides information on the operation and programming of the High Speed Paper Tape Reader, the High Speed Paper Tape Punch, and the Combination High Speed Reader/Punch. Included in this section are a general description, a table of status and command bytes, and sample programs for each device.

The above products all include a single-board device controller. Note, that with the Combination Reader/Punch, since there is only one device controller, the devices cannot be used simultaneously. To read and punch tapes at the same time, it is necessary to use products which would provide separate device controllers for each device.

8.4.2 General Description

Table 8-4 lists general characteristics of the Reader, the Punch, and the Controller.

TABLE 8-4.
READER AND PUNCH CHARACTERISTICS

Characteristics	Reader	Punch
Type	Photo-electric	Electro-mechanical
Tape Width	adjustable tape guides for 1/2', 11/16", and 1" tape	fixed width of 1"
Speed	maximum of 300 characters-per-second	maximum of 63.3 characters-per-second
Tape handling	paper, paper-mylar, and mylar	same as the Reader
Stop time	capable of stopping on a character (approximately 1 millisecond)	will punch the next character and stop

TABLE 8-4.
READER AND PUNCH CHARACTERISTICS (Continued)

Characteristics	Reader	Punch
Read/Load Lever	allows loading or changing of tapes of varying widths	does not apply
Power Switch	applies AC power to Reader motor	applies AC power to Punch motor
Remote Switch	does not apply	puts the Punch on-line with the Processor
Dimension	Rack Adapter - 19"W, 7" H, 10-1/4"D, 3/8" T	Punch - 8" W, 11"H, 16-1/2" D Rack - 19" W, 11-1/4" H, 22" D Front - 19" W, 15.72" H
Rack Mountable	Yes	Yes
Weight	Reader 15 lbs. Adapter 5 lbs.	Punch 24.5 lbs. Punch, Chassis (Incl. Pwr. Sup.) 33 lbs.
Power Requirement	115 VAC 50/60 Hz 150 watts	115 VAC 60 Hz Motor - 9A START " 2A RUN 50 Hz models are available Punch Magnets - 2A

Controller

INTERDATA Part No. 32-136F02

Size - Single 10 inch INTERDATA Standard wire wrapped circuit board

Back Panel Slots - requires two standard expansion slots

Weight - 1-1/2 pounds (approx.)

Power Req. - +5 VDC .75A
+16.5 .005A

8.4.3 Status and Command

Table 8-5. provides Status and Command Byte Data for the HSPTR/P.

TABLE 8-5.
READER/PUNCH STATUS AND COMMAND BYTE FORMAT

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	OV			NMTN	BSY	EX		DU
COMMAND BYTE	DISABLE	ENABLE	STOP	RUN	INCR	SLEW	WRITE	READ

STATUS BIT DESCRIPTIONS

Bit	Reader	Punch
OV	The Overflow bit is set when the Buffer Register is loaded from the Reader before the previous character has been transferred. This condition can only happen in the SLEW mode.	The Overflow bit is always in a reset condition in the WRITE mode.
NMTN	The No-Motion bit is set when the Reader has been issued a STOP command and the tape has stopped on the next character.	The No-Motion bit is always in a reset condition in the WRITE mode.
BSY	The BUSY bit is set when the Buffer Register is empty, waiting for a character from the Reader.	The BUSY bit is set when the Buffer is full, waiting to transfer to the Punch.
EX	The Examine bit is set whenever OV=1 or NMTN=1.	The Examine bit is always reset in the WRITE mode.
DU	The Device Unavailable bit is set when the power to the Reader motor is off, or the Read/Load lever is in the Load position (straight up).	The Device Unavailable bit is set when the power to the Punch motor is off, or the REMOTE switch is released, or a low tape

TABLE 8-5.
READER/PUNCH STATUS AND COMMAND BYTE FORMAT (Continued)

STATUS BIT DESCRIPTIONS (Continued)

Bit	Reader	Punch
DU (cont'd)		condition exists on the tape reel. There is no low tape sensor on the fan fold bins.

COMMAND BIT DESCRIPTIONS

DISABLE	This command inhibits Interrupts from the Device Controller from interrupting the Processor.	Same as the Reader.
ENABLE	This command permits Interrupts from the Device Controller to interrupt the Processor.	Same as the Reader
STOP	This command halts the motion of the tape after the next character has been read. The next character to be read is positioned over the sense lights when the tape stops.	This command halts the tape after the next character is punched. The Punch motor is also turned off, unless the Power Switch on the panel is depressed.
RUN	This command starts the tape moving and leaves the controller in the RUN mode.	This commands starts the Punch motor, unless the REMOTE Switch on the panel is released.
INCR	In this mode of operation, the tape is advanced one character when the controller is in the RUN mode and BSY=1. The tape stops after reading one character. The tape remains stopped until a Read Data instruction is issued by the Processor, which will reset BSY and start the tape moving.	Not used.

TABLE 8-5.
READER/PUNCH STATUS AND COMMAND BYTE FORMAT (Continued)

COMMAND BIT DESCRIPTIONS (Continued)

Bit	Reader	Punch
SLEW	In this mode of operation, the tape is advanced, reading continuous characters, until stopped.	Not used.
WRITE		Designates the High Speed Paper Tape Punch.
READ	Designates the High Speed Paper Tape Reader.	

8.4.4 Interrupts

When enabled in the READ Mode, the device controller generates an external device interrupt when a data character is present in the controller, waiting to be transferred to the Processor. When enabled, in the WRITE Mode, the device controller generates an external device interrupt when the controller is ready for another character to be punched.

8.4.5 Initialization

When the CLR switch on the Processor is depressed, the following occurs:

1. Interrupts of all kinds are disabled.
2. The BSY, NMTN, and EX status bits are set.
3. The DISABLE, STOP, INCR, and READ command functions are set.
4. The punch power is turned off unless the POWER pushbutton is depressed (locked on).

8.4.6 Punch Power Controls

There are two pushbuttons on the front panel of the High Speed Paper Tape Punch. The top button, labeled REMOTE, determines the state of the Punch in reference to the Processor. If this pushbutton is released, the Punch is off-line with the Processor and is said to be in a LOCAL Mode. If the button is depressed, the Punch is on-line with the Processor and is said to be in a REMOTE Mode. The POWER pushbutton is located directly below the REMOTE pushbutton. With the POWER Switch released, and the REMOTE Switch depressed,

Punch power may be turned on by the program via the command RUN bit. The power may also be turned off by the program via the command STOP bit. Figure 8-2 illustrates when the power is on/off as a function of programmed and manual controls. The letter B represents the POWER button depressed, $\overline{B}$ means the POWER button is released. The letter P represents program-controlled power on. $\overline{P}$ means program-controlled power off.

	$\overline{B}\ \overline{P}$	$\overline{B}\ P$	$B\ P$	$B\ \overline{P}$
REMOTE	OFF	ON	ON	ON
LOCAL	OFF	OFF	ON	ON

Figure 8-2. Punch Power

8.4.7 Mode Switching

With a Combination Reader/Punch, care must be used when switching modes. The following is an example of switching from the WRITE to the READ Mode.

```
.
.
.
.
WD                  WRITE DATA
.
.
.
.
SS                  WAIT FOR NON-BUSY
B       *-1,NZ
LI      READCM      SET READ MODE
OC
.
.
.
.
```

The sense loop is required to insure that the last character in the buffer register is punched prior to issuing the Output Command READ. If the READ command is given too soon, the last character is interfered with. This is because the command READ causes the character on the tape, under the sense lights to be strobed into the buffer register. The logic behind this is that when the Reader has been issued a command STOP (Output Command WRITE causes a stop action also), the tape stops with the next character to be read under the sense lights. Thus, a RUN/STOP action will not cause a skipping of characters on

tape. Because of this feature, there is no need to sense status and check for BSY = 1 in switching from the READ mode to the WRITE Mode.

Table 8-6 shows a sample program for the Combination Reader/Punch which reads a block of characters and punches a block of character.

TABLE 8-6.
SAMPLE PROGRAM FOR COMBINATION READER/PUNCH

```
*SAMPLE PROGRAM USING THE HSPTR/HSPTP IN MODE SWITCHING

*READ AND PUNCH 100 BYTES (REPRODUCE)

START   LI      BUFFER-1        RESET BUFFER PTR.
        ST      ADDR+1          ADDRESS DEVICE
        LI      3
        ADR
        LI      -100            SET BYTE COUNT
        ST      COUNT

INPUT   LI      '99'            READ MODE
        OC

INP1    RDS                     READ AND STORE
        B       ERROR
        ST      ADDR, I
        ISZ     COUNT           CHECK IF DONE
        B       INP1

OUTPUT  LI      BUFFER-1        RESET BUFFER POINTER
        ST      ADDR+1
        LI      -100            SET LINE COUNT
        ST      COUNT
        LI      '92'            WRITE MODE
        OC
```

TABLE 8-6.
SAMPLE PROGRAM FOR COMBINATION READER/PUNCH (Continued)

```
OUT1    L       ADDR, I     GET BYTE
        WDS                 WRITE
        B       ERROR
        ISZ     COUNT       DONE?
        B       OUT1        NO
WAIT    SS                  WAIT TIL LAST CHAR PUNCHED
        B       *-1, NZ
        LI      '20'        TURN OFF PUNCH
        OC
        B       INPUT       LOOP
BUFFER  DS      100
ADDR    DC      @BUFFER-1   AUTOINDEXED BUFFER PTR.
```

8.4.8 Device Number

The High Speed Reader/Punch is normally assigned address X'03' if using a Reader only. If using a Punch only, or both a Reader and a Punch, address X'13' is normally assigned. These device numbers are easily changed by a minor modification to the device controllers.

8.5 CARD READER OPERATION AND PROGRAMMING

8.5.1 General Description

The Card Reader employs a photoelectric read station and a vacuum throat feed assembly. A special "wide strobe" read technique is used to preclude loss of data, even on cards which have been mispunched by as much as plus or minus one-half column.

The card read rate is in excess of 200 cards per minute with a 500 card capacity for both the input hopper and the output stacker.

Throughout the read operation light current checks, dark current checks, and card motion checks are continuously performed to verify the performance of the Card Reader.

Card Reader

Dimensions: 13"H, 12"D, 23"W

Weight: 75 lbs.

Power Requirement: 115VAC, 300VA max.

Controller

INTERDATA Part Number - 32-084

Size - Single 10 inch INTERDATA standard wire wrapped Circuit Board

Back Panel Slots - Requires two standard expansion slots

Power Requirement - +5VDC .75A

Weight - 1.5 pounds (approx.) (M.B.)
1 pound (approx.) (cable)

8.5.2 Operator Controls

8.5.2.1 POWER

The lighted POWER pushbutton applies AC power to all circuits. The pushbutton is lit when the power is on.

8.5.2.2 MOTOR Start

The lighted MOTOR pushbutton starts the drive motor is no error indicator lights are lit. The pushbutton is lit when the drive motor is running.

8.5.2.3 Read START

The lighted START pushbutton clears all error indicators and advances the Card Reader to the "ready" state to begin a read cycle upon receipt of the proper signal. The pushbutton is lit when the switch is depressed and no errors have been detected.

8.5.2.4 Read STOP

The lighted STOP pushbutton inhibits further read cycles until Read START is again depressed. Read STOP action is delayed until the current read cycle is completed. The pushbutton is lit when the switch is depressed, or if the Card Reader is stopped due to an error detection.

8.5.3 Status Indicator Lights

8.5.3.1 POWER On

The indicator on the POWER Switch is illuminated when power is applied to the Card Reader.

8.5.3.2 MOTOR On

The MOTOR Switch indicator is illuminated when the motor is running.

8.5.3.3 Read START

The START Switch is illuminated when the switch is depressed and no malfunctions have been detected.

8.5.3.4 Read STOP

The STOP Switch is illuminated when the switch is depressed or the Card Reader has stopped due to a trouble detection, as described in the following paragraphs.

8.5.3.5 PICK FAIL

If a card fails to be picked upon command, the PICK FAIL indicator is illuminated.

8.5.3.6 CARD MOTION Error

If the interval between the time the selected card enters the read station and the time the card leaves, does not correspond to 85 + 1/3 columns (the total card width), the CARD MOTION indicator illuminates.

8.5.3.7 LIGHT CURRENT Error

When all photo-read-cells do not conduct whenever a card is not in the read station, the LIGHT CURRENT indicator illuminates.

8.5.3.8 DARK CURRENT Error

The DARK CURRENT indicator illuminates if all photo-read-cells do not go dark for some instant between the beginning of the card and column 1, or between column 80 and the end of the card.

8.5.4 Status and Command Bytes

Table 8-7 illustrates the status and command byte coding for the Card Reader.

8.5.5 Data Format

A card Feed command causes the card to move over the photo-read-cells column by column, starting with column 1. Every column read (blank columns are read as all bits zero) generates a data strobe for that column and initiates a data transfer cycle. The first Read Data instruction reads the top six rows of the column; the second Read Data instruction reads the bottom six rows of that column. Figure 8-3 is an example of the data byte format.

TABLE 8-7.
CARD READER STATUS AND COMMAND BYTE DATA
(HEX ADDRESS 04)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	EOV	TBL	HE	NMTN	BSY	EX	EOM	DU
COMMAND BYTE	DISABLE	ENABLE	FEED					

EOV — The EOV bit is set when the data is not taken from the Device Controller buffer before the next column of data arrives from the read station. This bit is reset by a FEED Command.

TBL/DU — These bits are set when the Card Reader fails to pick a card upon command, or when an error condition occurs in the Card Reader. The error conditions are:

1. Card Motion Error
2. Light Current Error
3. Dark Current Error

These error conditions prevent the reading of any more cards until manually reset by the operator.

HE — The HE bit is set when the last card in the input hopper has been read. When HE sets, NMTN is set. The HE bit must be manually reset by the operator.

NMTN — The NMTN is set except for the time between a FEED command and the time it takes for a card to pass through the read station.

BSY — The BSY bit is set while the Device Controller is awaiting data from the Card Reader. It resets when the data is available to be transferred.

EX — The EX bit sets when any one of the upper four (4) bits of the Status byte is set.

EOM — The EOM bit is set whenever NMTN is set, and when the input hopper becomes empty.

DISABLE — This command disables the Card Reader Device Interrupt.

ENABLE — This command enables the Card Reader Device Interrupt.

FEED — This command initiates a new card feed cycle; however, no action occurs if TBL, DU, or HE is set.

BIT NUMBER	0	1	2	3	4	5	6	7	
ROW NUMBER			12	11	0	1	2	3	FIRST DATA BYTE
ROW NUMBER			4	5	6	7	8	9	SECOND DATA BYTE

NOTE: Bit numbers 0 and 1 should always be zero.

Figure 8-3. Data Byte Format

8.5.6 Interrupts

When enabled (Bit 1 of the COMMAND byte set), the Card Reader Device Controller generates an external device interrupt for each column read. The interrupt indicates to the Processor that data is available for transfer.

8.5.7 Initialization

When CLR pushbutton on the Processor is depressed, the following occurs:

1. The NMTN and EOM bits are set.
2. The EOV bit is reset.
3. The BSY and EX bit are set.

8.5.8 Operator Procedures

After applying power to the Card Reader, allow it a few minutes to warm up. Cards should be placed face down in the hopper with the 12-edge toward the operator. Additional cards may be added to the hopper without interfering with the operation.

8.5.9 Programming

A sample card input routine is shown in Table 8-8. In the sample program, note that the HE bit (hopper empty) is checked before other bits. This bit does not become set until the last card is read. If 80 columns are not read from each card, there is a Card Reader malfunction, as all blank columns should be read as zeros.

Code conversion is required when reading conventional Hollerith cards.

See Table 8-9 of the Hollerith punched-card codes for the ASCII character set.

TABLE 8-8.
CARD READER SAMPLE SUBROUTINE

```
*READ 80 COLUMNS INTO 160 BYTE BUFFER

*CALL IS BAL READ

*

READ     DC     '0000'
         LI     BUFFER-1      RESET BUFFER POINTER
         ST     ADDR+1
         LI     -80           RESET COLUMN COUNT
         ST     COUNT
         LI     4             GET CARD DEVICE NUMBER
         ADR                  ADDRESS DEVICE

WAIT     SS                   SENSE STATUS
         SH     2, CO         GET EOM BIT INTO CARRY
         B      WAIT, NC      WAIT IF NO EOM
         NI     8             GET HE BIT
         B      EMPTY, NZ     EXIT IF HOPPER EMPTY

FEED     LI     'A0'          GET DEV COMMAND
         OC

READ 2   RDS                  READ ROWS 1-6
         B      ERROR         ERROR CONDITION EXIT
         ST     ADDR,I
         RD                   READ ROWS 7-12
         ST     ADDR, I
         ISZ    COUNT         TEST IF 80 COLS READ
         B      READ2         DO MORE
         B      READ, I       DONE

BUFFER   DS     160

ADDR     DC     @BUFFER-1     AUTOINDEXED POINTER
         END
```

TABLE 8-9.
ASCII TO CARD CODE CONVERSION

Graphic	8-Bit ASCII Code	7-Bit ASCII Code	Card Code	Graphic	8-Bit ASCII Code	7-Bit ASCII Code	Card Code
SPACE	A0	20	0-8-2	@	C0	40	8-4
!	A1	21	12-8-7	A	C1	41	12-1
"	A2	22	8-7	B	C2	42	12-2
#	A3	23	8-3	C	C3	43	12-3
$	A4	24	11-8-3	D	C4	44	12-4
%	A5	25	0-8-4	E	C5	45	12-5
&	A6	26	12	F	C6	46	12-6
'	A7	27	8-5	G	C7	47	12-7
(	A8	28	12-8-5	H	C8	48	12-8
)	A9	29	11-8-5	I	C9	49	12-9
*	AA	2A	11-8-4	J	CA	4A	11-1
+	AB	2B	12-8-6	K	CB	4B	11-2
,	AC	2C	0-8-3	L	CC	4C	11-3
-	AD	2D	11	M	CD	4D	11-4
.	AE	2E	12-8-3	N	CE	4E	11-5
/	AF	2F	0-1	O	CF	4F	11-6
0	B0	30	0	P	D0	50	11-7
1	B1	31	1	Q	D1	51	11-8
2	B2	32	2	R	D2	52	11-9
3	B3	33	3	S	D3	53	0-2
4	B4	34	4	T	D4	54	0-3
5	B5	35	5	U	D5	55	0-4
6	B6	36	6	V	D6	56	0-5
7	B7	37	7	W	D7	57	0-6
8	B8	38	8	X	D8	58	0-7
9	B9	39	9	Y	D9	59	0-8
:	BA	3A	8-2	Z	DA	5A	0-9
;	BB	3B	11-8-6	[	DB	5B	12-8-2
<	BC	3C	12-8-4	\	DC	5C	11-8-1
=	BD	3D	8-6	]	DD	5D	11-8-2
>	BE	3E	0-8-6	↑	DE	5E	11-8-7
?	BF	3F	0-8-7	←	DF	5F	0-8-5

CHAPTER 9

CONFIGURATION AND INSTALLATION PLANNING

9.1 INTRODUCTION

Modularity is the key word which describes the INTERDATA building block approach to configuring Model 1 Digital Systems. The highly modular structure of the Model 1 line of digital equipment permits custom configurations to suit the user's exact needs. It also provides the means for gracefully expanding a Model 1 Digital System as the user's requirements grow.

9.2 BASIC PROCESSOR CHASSIS

The basic Model 1 Processor is packaged in a single RETMA standard 5-1/4 inch rack mountable chassis. See Figure 9-1. This chassis includes a power supply, cooling fans, and room for 8 standard 10 inch printed circuit boards. The first 5 circuit board slots are used for the Processor and the first 2048 byte core memory module. The remaining 3 circuit board slots are universal expansion slots which can accept any combination (up to 3) of the following:

1. Additional 2048 byte core memory modules
2. 2048 byte read only memory modules
3. Standard INTERDATA peripheral device controllers
4. Standard INTERDATA Selector Channel
5. User designed interfaces to either the Multiplexor Bus (I/O Bus) or the Memory Bus (M Bus).

Wire wrapped circuit cards require two expansion card slot positions. Printed circuit cards require a single expansion card slot. The basic Model 1 chassis with 2048 bytes of memory and the standard control panel weighs about 42 pounds, including the power supply. Typical weight of circuit cards is approximately 1 pound.

9.3 EXPANSION CHASSIS

Model 1 system expansion chassis have identical dimensions as the basic Processor chassis. The expansion chassis contains 8 universal expansion slots which can accept any combination of up to 8 memory modules, peripheral controllers, selector channels, or user designed interfaces as described for the basic chassis. Expansion chassis include cooling fans and have room to accept an additional power supply, although it is not equipped with the power supply. Expansion chassis are supplied with two cables for connecting to the basic Processor chassis.

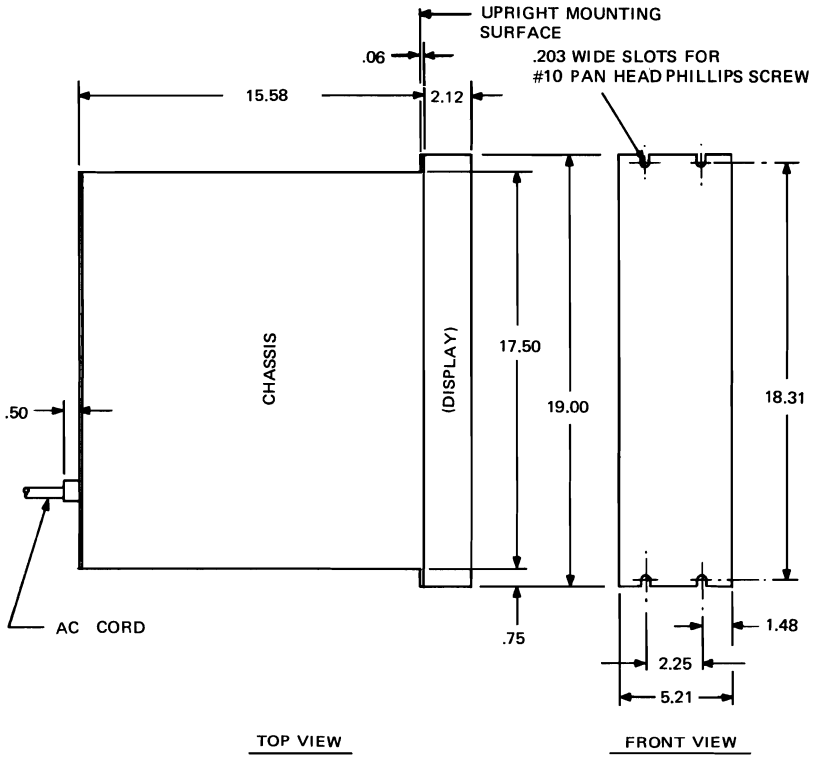


Figure 9-1. Model 1 Outline Drawing

Expansion chassis may be added to the Model 1 Digital System as required by the particular system configuration subject to certain constraints discussed later.

9.4 DECORATOR COVER

A decorator cabinet cover for a single 5-1/4 inch chassis is available for desk top mounting versions of the Model 1 Processor.

9.5 LINE POWER REQUIREMENTS

The basic Model 1 Processor requires 115VAC, ±10%, 47Hz to 63Hz, 4.5 amps maximum.

9.6 REGULATED POWER

The Model 1 power supply, included with the Processor, provides three regulated voltage supplies:

+5VAC ±1%	12 amps
+16VDC ±1%*	1.5 amps
-16VDC ±1%*	3.0 amps

and requires 115VAC, ±10%, 4.4 amps, 47Hz to 63Hz primary power.

9.7 CONFIGURING POWER REQUIREMENTS

The Model 1 Processor and the first 2048 byte memory module requires:

+5VDC	5.25 amps
+16VDC	1.15 amps
-16VDC	1.45 amps

which leaves

+5VDC	6.75 amps
+16VDC	.35 amps
-16VDC	1.55 amps

of regulated power for user designated expansions. Ordinarily, this is sufficient power for the basic Processor chassis plus one expansion chassis. If it is not, the user is expected to purchase an expansion power supply separately.

Table 9-1 lists the approximate regulated power requirements for typical Model 1 expansion modules.

TABLE 9-1
EXPANSION MODULES POWER REQUIREMENTS

Modules	+5VDC	+16VDC	-16VDC
2048 byte core memory	0.75	—	0.12
2048 byte read only memory	.80	—	.20
Selector Channel	1.0	—	
Typical printed circuit board	1.0	—	
Typical wire wrapped board	.80	—	

*The +16VDC and -16VDC supplies are temperature tracking supplies which vary, approximately linearly, from ±15VDC to ±18VDC over the temperature range 50°C to 0°C, respectively.

The exact power requirements for Model 1 expansions are listed in the documentation for each individual product.

9.8 CONFIGURATION CONSTRAINTS

Model 1 Digital Systems may be configured in an unusually wide variety of ways. However, the following factors should be considered when planning for the installation of a Model 1 Digital System.

1) Air Flow

At least two inches of free air space should be provided on each side, or at the rear, of each basic Model 1 chassis and each expansion chassis to permit adequate flow for air cooling. The Processor is designed to operate over an ambient temperature, ranging from 0 °C to 50 °C.

2) Expansion Chassis

Expansion chassis should be mounted either immediately above or immediately below the next chassis to minimize the lengths of the chassis to chassis interconnect cables.

3) Memory Modules

All memory modules must be physically located in the basic Processor chassis and in the first expansion chassis and all memory modules must be powered by the power supply in the basic processor's chassis.

4) Selector Channels, Universal Memory Bus Interfaces, Memory Bus

The Memory Bus may not extend beyond the first two expansion chassis which must be immediately above and below the basic Processor chassis. Any device physically connecting to the Memory Bus (i.e., Selector Channel, Universal Memory Bus Interface, or user designed interfaces) must, therefore, be physically located in one of these 3 chassis.

9.9 INSTALLATION

All necessary mounting hardware, interconnect cabling and installation instructions are shipped with the Model 1 Processor. Similarly mounting hardware, cabling, and installation instructions are provided with all standard INTERDATA peripheral device controllers. Installation of INTERDATA equipment by INTERDATA representatives can be arranged by contacting the local INTERDATA Sales Office.

CHAPTER 10

BASIC MODEL 1 PROGRAMMING

10.1 INTRODUCTION

The purpose of this chapter is to introduce the user/programmer to the Model 1. It provides helpful programming information and examples and is intended to aid the user who has little or no Model 1 programming experience. Some experience in assembly language programming is assumed. This document is designed to be used in conjunction with the Model 1 Assembler Language Specifications (see Chapter 11) and other software descriptions. A rigorous explanation of each Model 1 machine instruction is contained in Chapter 4. The reader should have a basic knowledge of the instruction set, and the symbolic assembly language before he attempts to study the programming examples given in this chapter.

10.1.1 The Model 1

The Model 1 Processor is designed around an eight-bit byte memory system. It is a page addressable machine with a minimum memory size of 2K bytes, and maximum of 16K bytes. Each page consists of 256 bytes (A 2K machine has eight pages). Instructions are of two lengths: one byte (short-form instruction) and two bytes (long-form instruction). Instructions may appear anywhere in memory without restrictions regarding byte or page boundaries. Addressing may be either direct or indirect. (See Chapter 3 and the Section on addressing techniques).

10.1.2 Programmable Registers

An eight-bit register called the Accumulator is available to the programmer. In addition, a Carry Bit associated with the Accumulator, is also programmable.

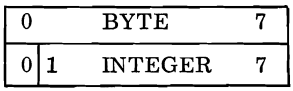
The Accumulator is involved in nearly all Model 1 instructions. The Carry Bit is linked to the Accumulator for arithmetic and shift operations, but it may also be independently controlled by the programmer.

10.1.3 Arithmetic/Logical Unit

The Arithmetic/Logical Unit processes binary integer and logical information. The operands are located in the accumulator and/or core memory. Arithmetic data is fixed point and is treated as a signed seven bit integer or as an unsigned eight-bit integer. Extended-precision arithmetic may be accomplished via software routines. Negative numbers are represented in 2's complement form with a sign bit of one. The numeric value of zero is always treated as non-negative. Table 10-1 shows several examples of the fixed-point number representation used in the Model 1.

TABLE 10-1
EXAMPLES OF FIXED-POINT NUMBER REPRESENTATION

NUMBER	DECIMAL	BINARY	HEXADECIMAL
2^7-1	127	0111 1111	7 F
2^0	1	0000 0001	0 1
0	0	0000 0000	0 0
$-(2^0)$	-1	1111 1111	F F
$-(2^7)$	-128	1000 0000	8 0
2^2+1	5	0000 0101	0 5



10.1.4 2'S Complement Notation

Negative numbers are represented in 2's complement notation. An eight-bit number is negative only if bit 0 is set. To change the sign of a number, the 2's complement of the number is produced in a two-step procedure:

1) Change all zeros to ones, and change all ones to zeros (complement every bit).

2) Add 1 to the number.

Example: The number five is represented in binary form as 00000101

Step 1) 11111010 (complement) = 1's complement of 5

Step 2) 11111011 (add one) = 2's complement of 5

The result is the 2's complement of 5 representing -5.

10.1.5 Hex Notation

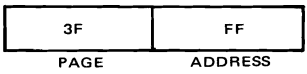
Binary information is expressed in hexadecimal notation (base 16) for purposes of simplicity. All references to binary instructions, data or addresses in Model 1 software are made in hex notation. Four binary bits of information can be expressed by a single hexadecimal character. An eight-bit byte of memory is expressed by a pair of hex characters. Table 10-2 lists binary, hexadecimal and decimal equivalents.

TABLE 10-2
HEXADECIMAL, BINARY, AND DECIMAL CROSS-REFERENCE

HEXADECIMAL	BINARY	DECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15
Example: The binary 2-byte number (0001111110100101) can be expressed in hex notation as '1FA5'		

10.2 ADDRESSING TECHNIQUES

The Model 1 is a page-addressed machine. The Memory System is subdivided into pages numbered in hex notation from '00' (page zero) to the top page of memory (which depends upon how many memory modules are attached to the machine). On a 2K system the pages are numbered from '00' to '07'. Adding another 2K, adds pages '08' to '0F'. On a full 16K system, the top page number is '3F'. Within each page there are 256 bytes numbered from '00' to 'FF'. A Model 1 address is made up of a page number combined with a byte location. The address length is 14 bits, and ranges from '0000' to '3FFF', as follows:



Memory reference instructions may directly address either the current page or page zero. The current page is the page on which the instruction itself lies. (If a two byte instruction lies in two pages, then the current page is the highest numbered page containing part of the instruction.) Thus a load instruction in page 4 may directly address locations '0400' to '04FF' and '0000' to '00FF'. Page 0 is an important "common" area that can be directly referenced from anywhere throughout memory.

Addressing other than the Current Page or Page 0 must be done Indirectly. Indirect addressing is accomplished as follows: a memory reference instruction (with the Indirect Bit Set) directly addresses a 2 byte address word in its resident page or page zero, and the least significant 14 bits of the 2 byte word becomes the effective address of the instruction.

Given below are examples of the four combinations of addressing on the Model 1.

1) Direct addressing of the current page:

```
        ORG  '0300'
        L    DATA     LOAD LOCATION DATA INTO A
        B    DONE
DATA    DC   7
```

2) Direct addressing of page 0.

```
        ORG  '0000'
DATA    DC   7

        ORG  '0300'
        L    DATA     LOAD LOCATION DATA INTO A
        B    DONE
```

3) Indirect addressing through the current page.

```
        ORG  '0300'
        L    ADDR, I  LOAD INDIRECT, LOCATION DATA
        B    DONE
ADDR    DC   DATA     ADDRESS CONSTANT FOR LOCA-
*                     TION DATA

        ORG  '0400'
DATA    DC   7
```

4) Indirect addressing through page zero.

```
          ORG  '0000'
ADDR      DC   DATA      ADDRESS CONSTANT OF DATA

          ORG  '0300'
          L    ADDR,I   LOAD INDIRECT, LOCATION DATA
          B    DONE

          ORG  '0400'
DATA      DC   7
```

In examples 3) and 4) "ADDR" is a two byte address word which points to location "DATA".

The symbolic address DONE always lies within page 3.

Programming note: Since direct addressing to a page other than the current page or page 0 is impossible, the Assembler Program will indicate an error during the assembly process if a referenced location is outside of the current page or page 0. In this case the assembler places an "A" flag next to the instruction to alert the programmer to the mistake.

EXAMPLE:

```
          ORG  '0300'
A         L    DATA      DIRECT ADDRESSING ATTEMPTED
          B    DONE      OUTSIDE OF CURRENT PAGE

          ORG  '0400'
DATA      DC   7
```

To rectify this condition, the programmer must set up an address constant and change the mode to indirect as shown in Example (3).

In writing programs for a paged machine, the difficulty of addressing locations outside the current page may arise. It is recommended that all coding be roughly pre-counted so that it can be divided into 240 byte blocks and placed on unique pages. Doing this allows the programmer to see what label references are outside of the current page (or page zero) so that he may re-arrange his coding or set up the necessary address constants for indirect addressing. Arranging coding in 240 byte blocks allows for 16 bytes of expansion in a page with little danger of page overflow.

10.3 MEMORY ORGANIZATION

Some areas of memory are dedicated to certain hardware and software functions. This section describes these areas, giving their purposes and specific locations.

Page 0, because of its universal addressing capability, is left mostly for the user. The locations in page 0 which have special uses by Model 1 software are shown in Table 10-3.

TABLE 10-3
TABLE OF COMMONLY USED ADDRESSES

LOC	USE	
'0000'	Power up restart point (Fixed hardware location)	
'000A'	Top of core pointer for the Text Editor	
'00F0'	Core available pointer for the Assembler	
'00FE' - '00FF'	Address constant used by DBUG for breakpoints	
'0100' - '0107'	Interrupt service block 0	FIXED HARDWARE LOCATIONS
'0108' - '010F'	Interrupt service block 1	
'0110' - '0117'	Interrupt service block 2	
'0118' - '011F'	Interrupt service block 3	
'0120' - '0127'	Interrupt service block 4	
'0128' - '012F'	Interrupt service block 5	
'0130' - '0137'	Interrupt service block 6	
'0138' - '013F'	Interrupt service block 7	
'0140'	Execute point for the Incore Loader	
'014A' - '014B'	Device and command for binary input	
'014C' - '014D'	Device and command for binary output	
'014E' - '014F'	Device and command for source input	
'0150' - '0151'	Device and command for list output	
'0152'	Incore loader page pointer	
'0154'	Entry point for binary input routine	
'017C'	Power up restart point (for Auto Control Panel)	
'0180'	Assembler Pass 1 start point	
'0182'	Assembler Pass 2 start point	
'0400'	Relocation pointer for the DBUG program	

NOTE

Systems with Auto Load Panel have the first 64 bytes of core frozen as read only memory.

Page 1 contains several dedicated locations:

1) Hardware Interrupt service table beginning at location '0100'. There are 8 bytes per interrupt line, and a maximum of 8 lines.

   The first four lines (numbers 0-3) are standard on the Model 1. The remaining lines are optional. Programming examples using the interrupt service blocks will be found in Section 4.

   Although these locations are dedicated to interrupt handling, each group of eight bytes may be used as ordinary memory locations if the respective interrupt is not being used.

2) Software In-Core loader and Device-Definition table.

   LOC '140'-'153'

   These locations contain the In-Core boot loader and a Device Definition table which allows the Model 1 software a certain degree of device independence. This area is described in the Loader Descriptions Section of Chapter 11. It should not be overwritten by the programmer unless absolutely needed, since the instructions contained in these locations will have to be manually entered if an initial load is required.

3) Software Binary Input Device driver.
   This set of coding is used by the In-Core Loader and the General Loader to drive the Binary Input Device. This driver is one of two forms:

   a) For High Speed Paper Tape Reader, Cassette Tape or parallel Teletype. LOC '154'-'160'

   b) For serial Teletype LOC '154'-'179'

   The driver areas also should not be overwritten by the programmer if the In-Core Loader or General Loader is to be used. The remaining locations of Page 1 and all other core locations are available to the programmer.

10.4 BASIC PROGRAMMING EXAMPLES

10.4.1 Moves

Move Field A to Field B (where fields are not in the same page as the move routine).

```
            ORG     '0200'
FIELDA      DS      10          TEN BYTE SOURCE FIELD
FIELDB      DS      10          TEN BYTE DESTINATION
                                FIELD
            ORG     '0300'
LOOP        L       SOURCE,I    LOAD FIELD A BYTE
                                INDIRECT
            ST      DEST,I      STORE FIELD B BYTE
                                INDIRECT
            ISZ     SOURCE+1    BUMP ADDRESS WORDS
                                BY ONE
            ISZ     DEST+1
            ISZ     CNT         BUMP LOOP COUNT
                                (SKIP IF TEN BYTES
                                DONE)
            B       LOOP        RETURN TO DO ANOTHER
                                BYTE
            B       DONE        FINISHED
CNT         DC      -10
SOURCE      DC      FIELDA      FIELD A ADDRESS WORD
DEST        DC      FIELDB      FIELD B ADDRESS WORD
```

This routine is not reusable as it now stands. The address words (SOURCE and DEST) and the loop count (CNT) must be re-initialized to their original values.

Move Field A to Field B (where fields are in the same page as the move coding).

```
*THE FOLLOWING CODING MAKES THE ROUTINE REUSABLE.
            ORG     '0200'
RESET       LI      -10
            ST      CNT         RESET LOOP COUNT
            LI      FIELDA      LOAD FIELD A LOCATION
```

```
            ST      LOAD+1      RESET SOURCE PTR
            LI      FIELDB      LOAD FIELD B LOCATION
            ST      STORE+1     RESET DESTINATION PTR
*           MOVE ROUTINE LOOP
LOAD        L       FIELDA      GET FIELD A BYTE
            ISZ     *-1         BUMP ADDRESS
STORE       ST      FIELDB      PUT FIELD B BYTE
            ISZ     *-1         BUMP ADDRESS
            ISZ     CNT         BUMP LOOP COUNT
                                (SKIP IF DONE)
            B       LOAD        RETURN TO DO NEXT
                                BYTE
            B       DONE        FINISHED
FIELDA      DS      10          TEN BYTE SOURCE FIELD
FIELDB      DS      10          TEN BYTE DESTINATION
                                FIELD
```

10.4.2 Multiple Precision Arithmetic (Triple Precision Example)

Add two 24-bit numbers (Add A to B, answer in B).

```
A       DS      3       FACTOR 1
B       DS      3       ANSWER HERE (FACTOR 2)
ADD     L       A+2     GET A
        A       B+2     ADD LEAST SIGNIFICANT BYTES
        ST      B+2     STORE IN B
        L       A+1     GET MIDDLE BYTE OF A
        TS      NC      SKIP IF NO CARRY FROM FIRST ADD
        ISZ     B+1     INCREMENT B+1 IF CARRY.  SKIP ADD
                        IF ZERO
        A       B+1     ADD MIDDLE BYTES
        ST      B+1     STORE IN B
        L       A       GET TOP A BYTE
        TS      NC      SKIP IF NO CARRY FROM MIDDLE ADD
        ISZ     B       BUMP B, SKIP IF ZERO
        A       B       ADD TOP BYTES
        ST      B       STORE IN B
        B       DONE    FINISHED
```

The present state of Carry is now the result of Carry from the entire 24-bit ADD.

10.4.3 Subroutine Linkage (Use of the BAL Instruction)

In the example below, a main program calls a subroutine (SUBR1) which calls another subroutine (SUBR2). The following sequence of execution occurs. Subroutine Coding examples are shown in Table 10-4.

1. BAL SUBR1 causes the address of RETRN to be deposited in the two byte area at SUBR1. Processing continues at SUBR1+2.
2. BAL SUBR2 causes the address of RETRN2 to be deposited in the two byte area at SUBR2. Processing continues at SUBR2+2.
3. B SUBR2, I causes the transfer of program control to the address at SUBR2, i.e. RETRN2, where processing resumes.
4. B SUBR1, I causes transfer to the contents of SUBR1, i.e. RETRN, where processing resumes.
5. Processing halts at DONE.

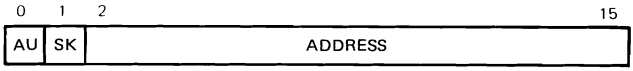
TABLE 10-4
SUBROUTINE CODING EXAMPLE

```
MAIN      ---
          ---
          BAL     SUBR1       LINK TO SUBR1
RETRN     ---
          ---
          ---
DONE      C       '02'        HALT
SUBR1     DC      '0000'      LOCATION COUNTER
          ---                 STORED HERE
          ---
          BAL     SUBR2       LINK TO SUBR2
RETRN2    ---
          ---
          B       SUBR1,I     RETURN FROM SUBR1
SUBR2     DC      '0000'      LOCATION COUNTER
          ---                 STORED HERE
          ---
          ---
          B       SUBR2,I     RETURN FROM SUBR2
```

10.4.4 Auto-Indexing/Auto-Skip

Auto-indexing is a feature on the Model 1 that allows automatic incrementing of an address word used in indirect addressing. Auto-skip is a feature that causes an automatic skip of the next 2 byte instruction if the resulting incremented byte location is not equal to 'FF'.

An address word is always 16 bits. The address itself comprises the low order 14 bits. Bit 0 is the Auto-indexing (AU) Bit, and Bit 1 is the Auto-skip (SK) Bit.



When using the Auto-indexing or the Auto-skip feature, the following operations take place:

1) When the AU Bit is set, the incrementing takes place on the byte-address portion of the address word (bits 8-15). It takes place anytime the address word is referenced by an indirect instruction. The incrementing takes place before the indirect instruction is executed, therefore if the programmer desires his first effective address be '0700', his address word should be '87FF'. The address increment will not overflow into the page portion of the address word.

2) When the Auto-skip flag is set, a test is made on the byte address portion of the address word after the indirect instruction has been executed. If it is not 'FF', then the location counter is incremented by 2. If it is 'FF', then the next sequential instruction is executed. The Auto-index and Auto-skip flags can be set on an address word by special prefixes in the assembler language or by address adjustment constants.

```
Example:    DC      @ADDR          SETS AUTO-INDEX FLAG
                                   ON ADDR

            DC      <ADDR          SETS AUTO-INDEX & SKIP
                                   FLAGS

            DC      ADDR+'8000'    SET AUTO-INDEX FLAG
```

Coding Example: Move page 7 to page 8 with instructions in page 2 using auto-indexing

```
        ORG   '0200'
LOOP    L     SOURCE,I    LOAD INDIRECT WITH
                          AUTO-INDEX
        ST    DEST,I      STORE INDIRECT WITH
                          AUTO-INDEX & SKIP
        B     DONE        FALL THRU IF '08FF'
                          REACHED
        B     LOOP        MORE
SOURCE  DC    '87FF'      AUTO-INDEX BIT SET
DEST    DC    'C8FF'      AUTO-INDEX & SKIP BIT
                          SET
```

Without the use of Auto-indexing this routine would require two more instructions (2 ISZ instructions for manually bumping the address words).

10.4.5 Condition Checks and Comparison

The following three hardware states can be tested on the Model 1.

1) The carry state
2) The sign of the accumulator
3) Zero condition of the accumulator

Tests for the negative condition of each of these states (NC no carry, NM non-minus, NZ non-zero) are available as modifier options on many of the Model 1 instructions.

There are 4 extended mnemonics available on the Model 1 Assembler in the form of compare instructions.

1) CEI - Compare equal immediate
2) CE - Compare equal
3) CLI - Compare low immediate (logical)
4) CL - Compare low (logical)

In these instructions, the accumulator is compared to the operand either immediate or memory reference) and if the condition is met (accumulator low or equal depending upon the mnemonic), the next instruction is executed. If the condition is not met then the location

counter is incremented by 2. To show this symbolically the following examples are given:

```
L     CHAR    |   L     RSULT
CEI   "A"     |   CL    DATA
B     EQUAL   |   B     LOW
B     NEQUAL  |   B     NOTLOW
```

NOTE: The CEI instruction is equivalent to the XI instruction with the N and NZ modifiers specified.

If the programmer desires to branch out on the not equal condition and fall through on equal, the following sequence may be used:

```
*     EXIT A ROUTINE IF ACCUMULATOR NOT EQUAL TO '12'
                *
                *
                *
XI              '12'
B               EXIT, NZ
                *
                *
                *
```

*NOTE: ACCUMULATOR DESTROYED IN THE ABOVE OPERATION

```
*     EXIT A ROUTINE IF ACCUMULATOR IS ZERO
                *
                *
                *
TS              NZ
B               EXIT
                *
                *
                *
```

10.4.6 I/O Programming

Several methods of I/O programming are available on the Model 1, I/O transfers can be accomplished in the Interrupt or Non-Interrupt mode, and through both serial and parallel transfers or Pulsed I/O. Examples and explanations of I/O programming are given in the following paragraphs.

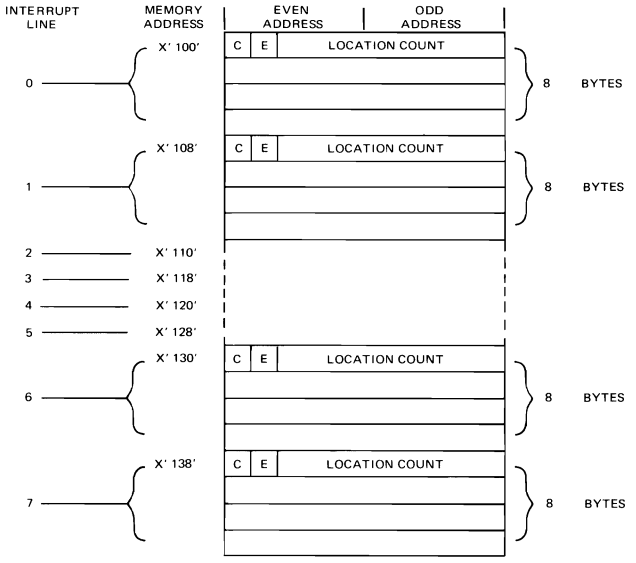
10.4.6.1 Parallel I/O

Parallel transfers are byte transfers made through the I/O Multiplexor Bus. An eight-bit byte or group of bytes is transferred to or from the Accumulator (data transfer) or core memory (block transfers). Four programming steps are required to complete a transfer.

1. Address the device using the ADR Instruction.
2. Send the proper device command to the device controller using the OC Instruction.
3. Wait until the device is ready to receive or send data. This can be accomplished by:
   a. The use of interrupts (the device interrupts the Processor when ready, i.e. Interrupt Mode).
   b. By sensing status on the device (wait for non-busy status to appear, i.e. Non-Interrupt Mode).
4. Accept or send the data with a Read or Write Instruction.

10.4.6.2 Interrupt Mode

When working in this mode, the programmer addresses the device, enables the external interrupt state with the Command Instruction (C '08'), enables the individual device interrupt by choosing the appropriate device command and sets the Mask Register to enable the appropriate external interrupt line. He then can proceed with processing until he receives an external interrupt. Control is transferred to the Interrupt Service Block associated with the external interrupt. See Figure 10-1. The device status is checked and data is received or sent accordingly. Through buffering, I/O can be overlapped with processing to take full advantage of available processing time.



NOTE – MEMORY LOCATIONS BEYOND X' 120' ARE AVAILABLE WITHOUT RESTRICTION IF THE PROCESSOR DOES NOT HAVE THE OPTIONAL 4 INTERRUPT LINES (4:7)

– THE FIRST INSTRUCTION OF THE NEW PROGRAM CORRESPONDING TO INTERRUPT LINE 0 WOULD BE AT LOCATION X' 102'

Figure 10-1. Interrupt Service Table

10.4.6.3 Non-Interrupt Mode

In this method, the time between the command to the device and the actual transfer is spent in a status loop wating for the device to become ready. This method does not make the most efficient use of Processor time because of the unused time in the status loop. The Non-Interrupt Mode is much easier to program. When the device is ready for transfer, the status of the device becomes zero (non-busy and no error conditions). Status may be checked with the SS Instruction, but some Read and Write Instructions have a built-in status check such as RDS, WDS, and the block transfer instructions.

When processing in mixed mode (both Interrupt and Non-Interrupt), care must be taken to insure that interrupts are disabled when initializing or completing transfers from Non-Interrupt Mode Devices. This must be done to insure that a device does not become "unaddressed" by an Interrupt Service Routine.

EXAMPLE I: Parallel Non-Interrupt Mode

```
            * WRITE THE CHARACTER "A" ON THE TTY
BEGIN   LI      2           LOAD TTY DEVICE #.
        ADR                 ADDRESS THE DEVICE
        LI      '98'        LOAD WRITE MODE COMMAND
        OC                  ISSUE WRITE COMMAND TO TTY
        LI      "A"         GET "A" INTO ACCUMULATOR
        WDS                 WAIT FOR BUSY TO DROP THEN
                            WRITE
        B       ERROR
        B       DONE

*READ 256 BYTES FROM HSPTR INTO AREA CALLED "BUFFER"

        ORG     '0300'
BEGIN   LI      '13'        LOAD HSPTR DEVICE #
        ADR                 ADDRESS HSPTR
        LI      '99'        LOAD READ COMMAND
        OC                  ISSUE READ COMMAND
        RB      ADDR,I      READ BLOCK INDIRECT
        B       ERROR
        B       DONE

ADDR    DC      BUFFER      ADDRESS CONSTANT OF BUFFER
        ORG     '0400'

BUFFER  EQU     *           1 PAGE (256 BYTE) READ BUFFER

*READ CHARACTERS FROM A TTY AND STORE THEM
*UNTIL A CR (CARRIAGE RETURN) IS READ.

BEGIN   LI      2           TTY DEVICE #
        ADR
        LI      '94'        TTY READ COMMAND
        OC
```

```
MORE    RDS             READ BYTE
        B       ERROR
        ST      BUFFER  STORE IN BUFFER
        ISZ     *-1     BUMP STORE ADDRESS
        CEI     '0D'    COMPARE EQUAL TO CR
        B       DONE    YES, CR
        B       MORE    NO, GET ANOTHER

BUFFER  DS      10
```

EXAMPLE II: PARALLEL INTERRUPT MODE

```
    * INITIATE A READ FROM THE TELETYPE.
    * CONTINUE PROCESSING UNTIL THE CHARACTER IS NEEDED.
    * IF 1ST NON-ZERO CHARACTER IS NOT ALREADY READ, THEN
    * WAIT FOR READ TO BE COMPLETE

          ORG   '0400'

    BEGIN LI    2               TTY DEVICE NO.
          ADR
          LI    'A6'            COMMAND READ AND ENABLE
                                DEVICE INTERRUPT
          OC
          LI    '10'            LOAD EXTERNAL INTERRUPT
                                MASK (LINE 3)
          C     '28'            ENABLE MOD1 EXTERNAL INTER-
                                RUPT AND SET MASK REGISTER
    ----
    ----
    ----
    ----
    ----
    ----            INTERMEDIATE PROCESSING NOT SHOWN
    ----
    ----
    ----
    FLAG  LI      0,NZ          WAIT LOOP FOR READ COMPLE-
                                TION
          B       *-2

    ----
    ----
    ----            CONTINUED PROCESSING WITH CHARACTER
    ----            IN ACCUMULATOR
    ----
    ----
    ----
    ----
    ----    B       DONE

    * LOC '0118' ASSUMED TO BE EXTERNAL INTERRUPT SERVICE
    * BLOCK
```

```
        ORG     '0118'          INTERRUPT LINE 3
IOLINE  DC      '0000'          LOCATION COUNT STORED HERE
        ST      AREG+1          SAVE ACCUMULATOR
        AK                      ACKNOWLEDGE THE INTERRUPT
        XI      2               CHECK IF DEVICE NUMBER IS 2
        B       ERROR,NZ        INVALID INTERRUPT IF NOT 2.
        LI      2               ADDRESS DEVICE
        ADR
        RDS                     GET CHARACTER
        B       ERROR
        ST      ADDR,I          STORE INDIRECT AT FLAGE+1
AREG    LI      0               RESTORE ACCUMULATOR
        B       IOLINE,I        RETURN FROM INTERRUPT
ADDR    DC      FLAG+1          ADDRESS CONSTANT OF FLAG+1
```

If the character is ready during the intermediate processing, the interrupt occurs, and the character is read and placed in FLAG+1 and processing continues where it left off. When the processing reaches FLAG the LI will load the character into the accumulator, resulting in a NZ condition which will cause the Processor to skip the looping branch. If the Processor reaches FLAG before the character is read, it will remain in the 2 instruction loop until interrupted and the non-zero character is placed in FLAG+1 by the read routine.

10.4.6.4 Serial I/O

Serial I/O is a bit-by-bit I/O process accomplished through the Serial I/O Port of the Model 1. The Serial I/O Port is connected to the Carry Flag of the Processor. Two processor Instructions are available to complete data transfers.

```
C       '10'    Input Serial Port to Carry
C       '80'    Output Carry to Serial Port
```

Serial devices such as a Teletype may be attached to the Serial I/O Port. The technique for serial I/O programming is to time the bit period of the device being programmed and use appropriate C Instructions each time a bit transfer period is reached. This is illustrated below for serial input from a Teletype:

1. Test the Serial I/O Port repeatedly and wait for it to become zero. This indicates the start of a character.

2. Wait one and one half bit periods, using a software timer or the Model 1 one Millisecond Clock. This will position us into the middle of the first data bit period.

3. Input the Serial I/O Port to Carry, and shift the Carry Bit into a Data Byte.

4. Wait one bit period.

5. Repeat Steps 3 and 4 until eight data bits have been shifted into a byte. The result is a single Teletype character.

Examples of Input and Output coding sequences follow:

EXAMPLE I - SERIAL INPUT

Input a character from a TTY attached to the Serial I/O port.

NOTE

A TTY character period consists of one start-bit period where the I/O port goes to zero, followed by eight data-bit periods which comprise the desired byte character. A bit period is approximately 9 milliseconds in duration. Following the eight data-bits are at least two stop-bit periods (ignored by the driver) where I/O port is a one.

```
*INITIALIZATION ROUTINE
*THIS ROUTINE ECHOS CHAR BACK TO PAGE PRINTER
*(UNBLOCKED TTY)

BEGIN   C    '10',NC     WAIT FOR START PERIOD
        B    *-2
        C    '80'        REMOVE THIS INSTRUCTION FOR
                         BLOCKED (NON-PRINTING) INPUT.
        LI   -9
        ST   CHAR        SET UP COUNTER FOR 9 PERIODS
        AA               DOUBLE ACCUMULATOR TO -18

*THE FOLLOWING ROUTINE IS A SOFTWARE TIMER. THE FIRST
*TIME THROUGH (WITH AN ARGUMENT OF -18) IT COUNTS
*13.5 MILLESECONDS TO POSITION OURSELVES IN THE
*MIDDLE OF THE FIRST DATA-BIT PERIOD. FROM THEN ON
*WITH AN ARGUMENT OF -12, IT COUNTS 9 ms. (THE
*MIDDLE OF EACH FOLLOWING DATA-BIT PERIOD).
        COM ST   COUNT
            LI   9
            AO
            B    *-1, NZ
            ISZ  COUNT
            B    COM+2

*DATA BIT READY (TIME IS UP)
```

```
TIME    LI    0          GET BYTE FOR BUILDING
        C     '10'       GET DATA BIT
        C     '80'       REMOVE THIS INSTRUCTION
                         FOR NON-PRINTING INPUT
        ISN   CHAR       CHECK IF 8 DATA PERIODS
                         COMPLETE
        B     DONE       YES, EXIT WITH BYTE IN
                         ACCUMULATOR
        SH    1,CI       SHIFT CARRY INTO DATA BYTE
        ST    TIME+1     RESTORE IN SAVE LOCATION
        LI    -12        SET 9MS. TIMER ARGUMENT
        B     COM        GO TO TIMER
COUNT   DC    0
CHAR    DC    0
```

EXAMPLE II

SERIAL OUTPUT ROUTINE (Using the 1 Millisecond Clock as the Timer)

```
*BYTE TO BE OUTPUT LOCATED AT DATA+1

BEGIN   LI    -10
        ST    CHR        SET UP FOR 10 BIT PERIODS
        LI    -18        WAIT 18 MILLISECONDS BEFORE
                         SENDING START BIT
        RC               RESET CARRY AS START PERIOD

COMOUT  ST    COUNT      STORE # OF MS TO BE COUNTED
        LI    '40'       LOAD CLOCK ENABLE MASK
                         (LINE 1)
        C     '28'       ACTIVATE CLOCK, ENABLE
                         INTERRUPT

WAIT    B     *          LOCKING HALT TO WAIT OUT
                         INTERRUPTS
        B     WAIT
TIME    C     '80'       SEND CARRY TO SERIAL PORT
        ISZ   CHR
        B     DONE       CHECK IF DONE

DATA    LI    0          GET BYTE TO BE OUTPUT
        SC
        SH    1, CO, CI  SHIFT OUT A BIT AND SHIFT IN
        ST    DATA+1     RESTORE DATA BYTE
        LI    -9         WAIT 9 MILLISECONDS (ONE
                         BIT PERIOD)
        B     COMOUT     BEFORE NEXT WRITE

CHR     DC    0
```

```
*THE 1 MILLISECOND CLOCK INTERRUPT LINE ASSUMED TO
 BE LOCATION '0108'.

          ORG  '0108'       LINE 1
CLOCK     DC   '0000'       ADDRESS OF TIME STORED HERE
          ISZ  COUNT        DONE?
          B    CLOCK,I      RETURN TO LOCKING HALT
          B    TIME         TIME IS UP, PROCESS BIT

COUNT     DC   0
```

When interrupts are enabled, the One Millisecond Clock causes an interrupt from the (B *) Loop at Location 'WAIT'. When the count does not show zero, a return is made (B CLOCK, I) which returns us to the B *). When the interrupt occurs, interrupts are disabled, but the enable flag is deposited in Bit 1 of the location counter stored at CLOCK. When the (B CLOCK, I) instruction is executed, the enable flag in the address word at CLOCK causes interrupts to be enabled again, so that we can be re-interrupted from the branch loop (B *) every millisecond.

Alternately, the programmer could use the time between interrupts to handle other processing instead of waiting in the (B *) loop. This overlap method of processing is the most efficient. The Serial I/O Port can also be attached to a separate interrupt line to give an interrupt on each zero period.

10.4.7 Power Fail and Restart Programming

The Model 1 is equipped with functions to facilitate the handling of a Power Fail condition, and to enable a resumption of a Power Fail interrupted program. The Model 1 has a dedicated Power Fail Interrupt Line, a "Command Power Down" Instruction, and a dedicated location ('0000') where processing resumes when power is returned to the Processor. The Power Fail Interrupt is maskable, but cannot be Disabled. Considering the above functions, an example of a Power Fail/Restart sequence is given.

```
*HEX LOCATION '0000' IS DEDICATED TO "POWER UP"
*CONTROL

          ORG  '0000'
          B    *+2,I        BRANCH INDIRECT TO POWUP
          DC   POWUP        ADDRESS CONSTANT OF POWUP
```

```
*ASSUME INTERRUPT LINE 0 (HEX LOCATION '0100') IS THE
*POWER FAIL INTERRUPT LINE

          ORG  '0100'
POWDWN    DC   '0000'      LOCATION COUNTER PLACED
                           HERE WHEN POWER FAIL
                           OCCURS
          ST   POWUP+1     SAVE ACCUMULATOR
          C    '03'        COMMAND POWER DOWN AND
                           HALT
POWUP     LI   0           RESTORE ACCUMULATOR
          B    POWDWN,I    RETURN TO INTERRUPTED
                           PROGRAM
```

10.4.8 Bit Instructions

The bit-oriented instructions can be used to efficiently evaluate Boolean expressions where the parameters are represented by bits in memory. Although this can be done by using conventional machine instructions, a considerable amount of shifting and complementing of operands and storing of partial results is required, much of which is saved by using the bit instructions. The Bit instructions can result in significant savings when large numbers of bits are involved.

Since the bit instructions are memory referenced, auto-indexing and indirect addressing modes are available, allowing tables of data to be operated on efficiently.

As in any programming effort, the organization of the data can have a significant effect on efficiency. Carefully examine the expression to be evaluated and, if possible, structure the data for efficient solution.

The following example, demonstrates many of the bit operation features.

EXAMPLE: Solve the Boolean expression

$$X = \overline{[A\bar{B}\bar{C} + C\bar{D} + E]} \cdot F + \bar{A}B + G$$

Assume for convenience the bits representing the variables are contained in core location ALPHA.

First note that:

$$A\overline{BC} = \overline{[\bar{A} + B + C]}$$

Thus, we can take advantage of the 'OR'ing of selected bits in one instruction.

```
BUF     DC   '00'          TEMP STORAGE AREA
MASK    DC   '07'
START   LI   '04'          SELECT BIT C
        RC
        OB   ALPHA         CARRY=C
        AA                 SELECT D BY SHIFTING MASK
                           LEFT
        NB   ALPHA,BN      CARRY=C D̄
        AA
        OB   ALPHA         CARRY=C D̄ + E
        L    ALPHA
        XI   '01'          COMPLEMENT BIT A
        OB   MASK,BN       CARRY=AB̄C̄ + C D̄ + E, SEE
                           NOTE ABOVE
        LI   '20'          SELECT F
        NB   ALPHA,CN      CARRY= [AB̄C̄ + C D̄ + E] • F
        L    BUF
        SH   1,CI,CO       CAPTURE AND CLEAR CARRY
        ST   BUF           PREVIOUS STATE OF CARRY IS
                           IN BIT0 OF BUF
        LI   '01'          SELECT A
        OB   ALPHA         CARRY=A
        AA                 SELECT B
        NB   ALPHA,CN      CARRY=ĀB
        LI   '40'          SELECT G
        OB   ALPHA         CARRY=ĀB + G
        LI   'FF'
        OB   BUF           CARRY=SOLUTION
        END
```

Note that the last instruction collected all the partial sums (in this case only one) that were stored in BUF. If the expression contained more partial sums, they would have been evaluated separately, collected in BUF by shifting with Carry, then one final instruction completes the solution, as in the example.

If the expression happened to be in the form of partial products, i.e.

$$FNC = (X + X + X) \cdot (X + X) \cdot (X+X+X) \cdot X$$

collect the partial products in the BUF as in the above, complement the BUF contents, then execute an NB with BN to complete the solution. (Similar to the way $A\overline{BC}$ was evaluated in the example).

10.5 SOFTWARE AND PROGRAM USAGE

10.5.1 Model 1 Software Summary

A set of basic software is available to the programmer to enable efficient programming of the Model 1. Included are:

1) An Assembler for writing complete symbolic assembly language programs.

2) A Debug Program (called DBUG) used for interactive testing of programs.

3) A Text Editor (called TEDIT) used for preparing or modifying assembler source tapes.

4) An In-Core loader for system initialization

5) A General Loader for loading user and Model 1 programs in standard loader format.

6) An Unloader used for punching an object program from core memory onto tape in standard loader format.

All of the above programs will run on a minimum (2K byte) system with the exception of the assembler which requires 4K bytes.

For specific descriptions on all of the above programs, refer to Chapter 11.

10.5.2 Model 1 Tape Formats

There are 3 forms of standard program tapes that can be processed by Model 1 software:

1) Binary object tape

2) Standard loader format tape

3) Standard assembler source format tape.

A binary object tape is a 256 byte tape with one 8-bit byte per tape frame that can be read into a page of memory by the In-Core loader. The tape is core image for 256 bytes from the point where the read begins. Only the General Loader, Unloader and Memory Test are available in this format. No current Mod 1 software will produce this tape format.

Standard loader format is the object tape formate output by the assembler. It has 4 data bits (one hex character) per tape frame. The high-order bits are special zones devised to allow free use of a TTY as a punch device. Each tape record consists of a logical 8-bit checksum (2 frames), a 16 bit origin or end/transfer address (4 frames), and a

variable number of data bytes (2 frames per byte). Blank tape signifies the end of data. For more detailed information on the above 2 formats see the Loader Descriptions section in Chapter 11.

Standard assembler source format can be used for both assembly language source programs and user program input data. For assembly language programs each record may be a maximum of 67 ASCII characters and must be followed by a CR and 8 rubouts.

Table 10-5 shows the tape format type processed by Model 1 software and the form in which each program is available.

TABLE 10-5
TAPE FORMAT SUMMARY

PROGRAM	TAPE FORM	TAPE INPUT	TAPE OUTPUT
IN-CORE LDR		BINARY	
GENERAL LDR	BINARY	LOADER	
UNLOADER	BINARY		LOADER
ASSEMBLER	LOADER	SOURCE	LOADER
DBUG	LOADER		
TEXT EDITOR	LOADER	SOURCE	SOURCE
PROCESSOR TEST	LOADER		
MEMORY TEST	BINARY		

10.5.3 Loading Procedure and Core Usage

To load a Model 1 program that has been assembled by the Model 1 Assembler assuming Standard Control Panel:

1. Insure the correct Incore Loader is in memory.
2. Locate a page of memory (other than page 1) in which locations '80' through 'FF' will not be used by the program. (i.e. No data is to be loaded there)
3. Manually enter this page number into location '0152' with the Switches.
4. Load the Model 1 General Loader, and then load the user object program as described in the Loader Descriptions publication.

User written programs should be located in memory in such a way that they:

1. *Do not overlay the In-core loader in Page One.*
2. Do not overlay the top half of at least one page. This area (location '80' through 'FF') will be needed for the Model 1 General Loader.
3. Do not use locations '00FE' and '00FF' if the program is to be debugged using the breakpoint feature of the Model 1 debug program.

10.5.4 User Program Relocation In The Model 1

Because the Model 1 is a page-addressed machine, program relocation is not a trivial task. The simplest procedure would be to reassemble the program with a new starting location (ORG) on the desired pages. To relocate a Model 1 program anywhere in memory without reassembly is difficult because a program's direct addressing ability depends on a single block of coding being resident in a single page. But it certainly is feasible to consider "page" relocation, ie., a program originally assembled for pages 3,4,5 could be moved as a block to 6,7,8. This is a valid move, only if the address constants used in pages 3,4,5 (used to refer indirectly to each other) are adjusted upward by the proper bias value. In this case, a block move from pages 3,4,5 to pages 6,7,8 followed by adding 3 to the page address of every address constant in 3,4,5 (that refers to one of the other two pages) would suffice.

CHAPTER 11
MODEL 1 SOFTWARE

11.1 MODEL 1 ASSEMBLER

11.1.1 Introduction

The Model 1 is a programmable digital system. The flexibility provided by the stored program allows the hardware to solve a wide range of problems. A program stored in the Model 1 memory comprises binary coded instructions and data. The binary form of a program is referred to as an object program.

To make the process of programming faster and easier, the user can write his program using symbolic operation codes, operands, modifiers and labels for memory locations. When a program is in this form, it is referred to as a source program.

The translation from the symbolic source program to the binary object program is done by the assembler. The Assembler runs on all Model 1 Systems with 4K or more and a Teletype. Table 11-1 is an example of a typical source program that searches for the first occurrence of the number 15 in a 256 byte table.

TABLE 11-1.
TYPICAL SOURCE PROGRAM

·NAME(LABEL)	OPERATION	OPERAND	COMMENT
	ORG	'0100'	SET THE LOCATION COUNTER
BEGIN	LI	15	SEARCH VALUE OF 15
LOOP	CE	ADR, I	COMPARE INDIRECT OF TABLE
	B	FINI	MATCH FOUND
	ISZ	ADR+1	NO MATCH, BUMP ADDRESS
	B	LOOP	GO LOOK AT NEXT BYTE
FINI	C	'02'	HALT PROGRAM
ADR	DC	TABLE	INDIRECT ADDRESS WORD FOR TABLE
	ORG	'0200'	
TABLE	DS	'0100'	
	END		

An assembly is performed in two passes. This means that the source tape or cards must be read twice. On the first pass, as the statements are read, a symbol table is built. This table contains the definition of every symbol encountered in the program. A symbol may be used as a name or an operand. The Symbol table is printed at the end of the Pass 1 in alphanumeric order. See Table 11-2. On the second pass, object code is generated and a listing is produced. As the object code for each statement is generated, it is stored in a 128 byte buffer. The contents of this buffer are punched when it becomes full or the origin of code generation is changed or an END statement is read.

The listing is a printed record showing each source statement and the binary data generated for that statement. Binary information is represented in hexadecimal form.

TABLE 11-2.
TYPICAL SYMBOL TABLE

Location	Symbol
010C	ADR
0100	BEGIN
010A	FINI
0102	LOOP
0200	TABLE

11.1.2 Assembly Listing

The Assembly listing is produced as part of the assembly process. The listing contains the source statements and the data generated from each statement.

The first four hexadecimal digits represent the value of the location counter. The next 2 or 4 hex digits represent the data generated by the assembler from the source statement. Table 11-3 shows the assembly listing for the previous example. Error flags may precede the location counter values. These flags indicate that an error was encountered in interpreting the statement. The meaning of each flag is as follows:

F - format error (invalid modifiers or operand for a given op-code)

M - multiple defined symbol (same symbol appears twice in label field)

O - op-code error (invalid mnemonic op-code)

U - Undefined symbol (symbol appears in operand field but never in label field)

A - address error (direct addressing was attempted outside of current page or page zero)

TABLE 11-3.
TYPICAL ASSEMBLY LISTING

LOCATION	DATA	NAME	OPERATION	OPERAND	COMMENT
			ORG	0100	SET LOCATION COUNTER
0100	900F	BEGIN	LI	15	SEARCH VALUE OF 15
0102	F70C	LOOP	CE	ADR,I	COMPARE INDIRECT TO TABLE
0104	610A		B	FINI	MATCH FOUND
0106	D50D		ISZ	ADR+1	NO MATCH, BUMP ADDRESS
0108	6102		B	LOOP	GO LOOK AT NEXT BYTE
010A	8002	FINI	C	'02'	HALT PROGRAM
010C	0200	ADR	DC	TABLE	ADDRESS INDEX CONSTANT
			ORG	'0200'	
0200		TABLE	DS	'0100'	
0300			END		

Whenever an F or O error occurs, the assembler advances the location counter by 2 bytes so that the program can be patched easily for debugging without re-assembly.

The symbol table that is built during the assembly process is printed following the END assembly pseudo-op at the end of pass one. The symbols are listed alphabetically with their values.

Preceeding each symbol in the symbol table is a field for error flags. These flags are as follows:

U - undefined symbol

M - multiple defined symbol

If the symbol is undefined, the last value of the location counter for a statement referencing the undefined symbol is printed.

11.1.3 The Assembler Language

11.1.3.1 Source Statements

There are two basic kinds of source statements, instruction statements and comment statements. Instruction statements are used for machine instructions and assembler instructions.

Comment statements are used for narrative explanation. They begin with *, and should not be confused with the comments of the instruction statement. Comment statements can occupy the entire statement line.

11.1.3.1.1 Instruction Statements

The comment and instruction statements are written by the programmer on a coding form that has the various fields clearly marked. This form, when filled out, is used to generate the source paper tape or source cards that are read by the assembler during the assembly process. On a coding form, the Name (label) begins in column 1, the Operation begins in column 10 and the Operand begins in column 16. The fixed field positions are a convenience for the programmer only, and are not required by the assembler. The assembler simply requires that fields be separated by one or more spaces. The fields are described in the following paragraphs.

NAME (LABEL)

A name is from one to six characters in length. The name must be written with the first character in column 1, and it must not contain any blanks. Names are used by the programmer to identify data and instructions in the program. The first character must be a letter; the remaining five can be letters or numbers. Typical names are:

NAME (LABEL)	OPERATION
START	
ARG1	
LOOP2	
GO	

OPERATION

The operation field specifies a machine instruction that is translated by the assembler to machine code, or it specifies an assembler instruction (pseudo-op) to control the assembly process. An operation is always required in an instruction statement, and should be written on the coding form beginning in column 10. No blanks may be used within the operation. Typical operations are:

NAME (LABEL)	OPERATION	OPERAND
	ORG	
	LI	
	A	
	DC	

In addition to the machine instruction set, there are 4 extended mnemonics (compare instructions) designed for programming ease. See 10.4.5 for their explanation and use.

OPERAND/MODIFIER

Operands identify the data to be used by the instruction. Some instructions require operands, e.g. Load, Add, Subtract. Only one operand per instruction is valid. These operands refer to Names (Labels) or constants. Other instructions do not have operands e.g. Test & Skip, Read Data. Modifiers are special mnemonics which add conditional or special functions to certain instructions, e.g., NZ = non-zero, I = indirect. The number and type of modifiers allowed depends on the operation. Modifiers are strung after the operand (if one exists) and are separated by commas. Modifiers may appear in any order. Typical operands and modifiers are:

NAME (LABEL)	OPERATION	OPERAND/MODIFIER
	A	TEMP
	B	OUT
	ST	TEMP, I
	B	IN, NZ, I
	TS	NC, NM

Valid modifiers for each instruction are explained in chapter 4, and summarized in Table 11-4. No spaces may appear between operands or modifiers.

TABLE 11-4.
SUMMARY OF INSTRUCTIONS

Mneumonic	Instruction	Instruction Type	Number of Operands	Valid Modifiers
A	ADD	MEM REF	1	I, N
AA	ADD A TO A	SHORT FORM	0	NC, NZ, NM
ADR	ADDRESS	SHORT FORM	0	
AI	ADD IMMEDIATE	IMMEDIATE	1	N, NC, NM
AK	ACKNOWLEDGE	SHORT FORM	0	
AO	ADD 1 TO A	SHORT FORM	0	NC, NZ, NM
B	BRANCH	MEM REF	1	I, NC, NZ, NM
BAL	BRANCH AND LINK	MEM REF	1	I, NC, NZ, NM
C	COMMAND	IMMEDIATE	1	NC, NM
CA	COMPLEMENT A	SHORT FORM	0	NC, NZ, NM
CC	COMPLEMENT CARRY	SHORT FORM	0	NC
CE	COMPARE EQUAL	MEM REF	1	I
CEI	COMPARE EQUAL IMMEDIATE	IMMEDIATE	1	
CL	COMPARE LOW	MEM REF	1	I
CLI	COMPARE LOW IMMEDIATE	IMMEDIATE	1	
CLR	CLEAR A	SHORT FORM	0	NC, NZ, NM
DC	DEFINE CONSTANT	PSEUDO-OP	1-n	
DS	DEFINE STORAGE	PSEUDO-OP	1	
END	END SOURCE	PSEUDO-OP	0-1	
EQU	EQUATE	PSEUDO-OP	1	
ISN	INCREMENT & SKIP NZ	MEM REF	1	I
ISZ	INCREMENT & SKIP Z	MEM REF	1	I
L	LOAD	MEM REF	1	I
LI	LOAD IMMEDIATE	IMMEDIATE	1	NZ, NM, N
N	AND	MEM REF	1	I, N
NB	AND BIT	MEM REF	1	I, BN, CN
NI	AND IMMEDIATE	IMMEDIATE	1	N, NZ, NM
NOP	NO-OPERATION	SHORT FORM	0	
O	OR	MEM REF	1	I, N
OB	OR-BIT	MEM REF	1	I, BN, CN
OC	OUTPUT COMMAND	SHORT FORM	0	
OI	OR-IMMEDIATE	IMMEDIATE	1	N, NM, NZ
ORG	ORIGIN	PSEUDO-OP	1	
PIO	PULSED I/O	SHORT FORM	0	1, 2, 3
RB	READ BLOCK	MEM REF	1	I
RC	RESET CARRY	SHORT FORM	0	NC
RD	READ DATA	SHORT FORM	0	

TABLE 11-4.
SUMMARY OF INSTRUCTIONS (Continued)

Mneumonic	Instruction	Instruction Type	Number of Operands	Valid Modifiers
RDS	READ DATA & SKIP	SHORT FORM	0	
RT	ROTATE	IMMEDIATE	0	CO, NC, NM, 1-8
S	SUBTRACT	MEM REF	1	I, N
SC	SET CARRY	SHORT FORM	0	NC
SH	SHIFT	IMMEDIATE	0	CO, CI, 1-8, NC, NM
SI	SUBTACT IMMEDIATE	IMMEDIATE	1	N, NM, NC
SS	SENSE STATUS	SHORT FORM	0	
ST	STORE	MEM REF	1	I
TS	TEST AND SKIP	SHORT FORM	0	NZ, NC, NM
WB	WRITE BLOCK	MEM REF	1	I
WD	WRITE DATA	SHORT FORM	0	
WDS	WRITE DATA & SKIP	SHORT FORM	0	
X	EXCLUSIVE-OR	MEM REF	1	I, N
XI	EXLUSIVE OR IMMEDIATE	IMMEDIATE	1	N, NM, NZ

MODIFIER DESCRIPTION

N	INHIBIT A-REG. ALTERATION
NC	SKIP ON NOT CARRY
NM	SKIP ON NOT MINUS
NZ	SKIP ON NOT ZERO
I	INDIRECT
CI	CARRY IN
CO	CARRY OUT
BN	BIT NOT
CN	CARRY NOT

COMMENT

Comments are descriptive text. Comments are printed on the assembly listing, along with the name, operation, and operand of the source statement. Comments are written beginning after the first blank in the operand, and can continue to column 71 of the coding form. Typical comments are:

OPERATION	OPERAND	COMMENT
L	TEMP, I	FETCH FIRST VALUE
ST	START	TABLE AREA
B	STOP	ERROR STOP

11.1.3.1.2 Comment Statements

Comment statements are descriptive text that can occupy the entire source statement line. Comment statements are written with an asterisk (*) in column 1, followed by any descriptive text the programmer desires. Any number of comment statements may be used at any place in a program. Comment statements do not produce binary object information and are used only as documenting aids. Typical comment statements are:

```
*  THIS IS A COMMENT STATEMENT, IT
*  CAN BE USED ANYWHERE IN A
*  PROGRAM AS A PROGRAMMER'S COMMENT
*  AND DOCUMENTATION AID
```

11.1.3.1.3 Character Set

All source statements are written using the following characters:

Alphabetics	A through Z
Numerics	0 through 9
Special characters	<+ - , = * ' (@ " and blank

11.1.3.2 Instruction Statement Format

The source instruction statement consists of:

- a name
- an operation
- an operand/modifier
- a comment

Each entry in a source statement may be composed of one or more items depending on the kind of source statement begin written.

- a name, when present, must be a symbol
- an operation, always present, must be a machine instruction mnemonic or a pseudo instruction mnemonic.
- an operand when present, may be composed of one or more expressions, which in turn, are composed of symbols, constants and arithmetic combinations of symbols and constants.
- a modifier must be a mnemonic valid for the operation specified.

11.1.3.2.1 Symbols

A symbol is used as a name or as an operand. In either case, symbols consist of from one to six characters. The first character must be alphabetic. The characters that can be used for a symbol are:

Alphabetics	A through Z
Numerics	0 through 9

The following symbols are valid and could be used as a name or as an operand.

T2

LOOP25

N

STOP

The following symbols are invalid for the reasons given:

2TOP	First character is not alphabetic
COMMAND	More than 6 characters
A TO D	Contains a blank
X4.2	Contains a special character, a period

11.1.3.2.2 Assembler Constants

Assembler constants appear as an operand for both machine instructions and assembler instructions. A constant can be one of 4 types:

- Address
- Decimal
- Hexadecimal
- Character

An address constant is assumed if the first character is alphabetic. If the first character is numeric, a decimal constant is assumed. Constants are also identified by a delimiter.

Delimeter	Constant Type
@	Address (with auto-indexing)
<	Address (with auto-index and auto skip)
'	Hexadecimal (single quote)
"	Character (double quote)
-	Decimal (minus)

Decimal constants always generate one byte of data and may vary from 0 to 255 in magnitude. A preceding minus sign (-) generates the 2's complement of the magnitude.

EXAMPLES:	SOURCE	GENERATED OBJECT CODE
	0	00
	128	80
	-1	FF
	10	0A
	300	format error (magnitude too large)

Hexadecimal constants may contain any even number of hex characters contained in single quotes ('). The valid hex characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. One byte is generated for each pair of characters.

EXAMPLES:	SOURCE	OBJECT CODE
	'10FE'	10FE
	'103'	(format error: odd number of characters)
	'1G'	(format error: invalid hex character
	'0123456789'	0123 4567 89

Character constants may contain any number of characters (including blanks) enclosed by the double quote character ("). The Assembler generates the equivalent 7-bit ASCII code.

EXAMPLES:	SOURCE	OBJECT CODE
	"ABCDEC"	4142 4344 4543
	"MODEL 1"	4D4F 4445 4C20 31

Address constants always generate the 14-bit address of the symbol given. If the symbol is preceded by @ the auto-index bit (bit 0) of the address constant is set. If the symbol is preceded by < the

auto-index and auto-skip bits (bits 0 and 1) are both set. If the symbol is preceded by neither of the above, both bits remain reset.

EXAMPLES:	SOURCE	OBJECT CODE
	ADDR	0200
	@ADDR	8200
	<ADDR	C200

11.1.3.2.3 Expressions

An expression is a constant or a symbol followed by an optional arithmetic operator (+ or -) followed by a hex or decimal constant. Expressions can be used as address constants or as instruction operands. Examples of expressions are:

OPERAND

ADDR
'2000'
-96
LOOP+4
TABLE-'1020'
A+100

11.1.3.2.4 Location Counter

The value of the location counter can be referenced by using an "*", which means "current value of the location counter". Addressing relative to the location counter is on a byte basis. To specify an address that is 2 bytes forward, the correct expression is *+2.

11.1.4 Machine Instruction Format

The assembler provides the facility for representing all the machine instructions with mnemonics. The binary instruction is generated by the assembler from the operation mnemonic the operand and the modifier. Each instruction has a unique mnemonic, that is used as the operation. These mnemonics and their meanings are listed in Table 11-4 along with valid operands and modifiers.

11.1.4.1 Short Format Instructions

The short form (8-bit) instructions have no operands but may have optional modifiers.

11.1.4.2 Memory Referenced Instructions

All memory referenced instructions must have symbolic or real operands which refer to addresses within their current page or page zero. Address violations are flagged with an "A" in the assembly listing. This means the programmer attempted symbolic direct addressing outside the current page or page zero.

11.1.4.3 Immediate Instructions

Except for shift and rotate which take no operand, the immediate instruction operand may be any valid expression. The assembler allows the 8-bit literal to be specified in a variety of ways. The simplest is a hexadecimal constant, in which the instruction takes the form:

```
I    'n'
```

Where I is an operation mnemonic and 'n' is two or four hex digits. (An odd number of digits or more than four digits is flagged with an F in the assembly listing). If four hex digits are specified, the right most two digits are used to generate the eight-bit immediate operand.

It is convenient to use the immediate constant to represent an address. Memory addresses are 14 bits long and the immediate constant is only 8 bits. Therefore, provision is made to select either the high or low order byte of an address. The instruction takes the form:

```
I    SYMBOL
I    (SYMBOL
```

The left parentheses "(" specifies the high order byte of the address defined by SYMBOL. If the parenthesis is absent, the low order byte of the address is implied.

EXAMPLES:

	OPERATION	OPERAND/MODIFIERS
SHORT FORM WITH NO MODIFIER	RD	
SHORT FORM WITH MODIFIER	TS	NC
MEM REF WITH SYMBOLIC OPERAND	L	SYMBOL
MEM REF WITH HEX OPERAND AND MODIFIER	L	'12',I
MEM REF WITH DECIMAL OPERAND	ST	30
IMMEDIATE WITH DECIMAL OPERAND	LI	2
IMMEDIATE WITH HEX OPERAND AND MODIFIER	LI	'80',NZ
IMMEDIATE WITH SYMBOLIC OPERAND	AI	SYMBOL

11.1.5 PSEUDO–OPS

PSEUDO-OPS are used to control the assembly process, define symbols, and generate data. PSEUDO-OP statements do not always generate data as the machine instruction statements do. The following paragraphs describe the PSEUDO-OPS.

11.1.5.1 Symbol Definition

EQU-Equate Symbol

NAME	OPERATION	OPERAND
A Symbol required	EQU	An expression

The EQU statement is used to equate a symbol to the value of an expression. Symbols used in the expression must be previously defined.

NAME	OPERATION	OPERAND
LOOP	EQU	LOOP1
TOP	EQU	END-64
DELTA	EQU	BOTTOM
HERE	EQU	*
START	EQU	'01FE'
TWO	EQU	2

11.1.5.2 Data Definition

There are two data definition instructions; the DS and the DC. These PSUEDO-OPS provide a convenient means to define and reserve a data storage.

11.1.5.2.1 DC - Define Constant

NAME	OPERATION	OPERAND
A symbol (optional)	DC	One or more operands separated by commas

The DC assembler instruction is used to define constants. These constants may be hexadecimal, decimal, character, or address.

CONSTANT TYPE	MACHINE FORMAT
Character	8-bit character code
Hexadecimal	8-bit binary
Decimal	8-bit binary
Address	16-bit binary

11.1.5.2.1.1 Character Constant

The character constant can be any length. It must be enclosed in double quotation marks.

NAME	OPERATION	OPERAND
MESG1	DC	"LOAD THE TAPE"
	DC	"EXECUTE AT 19FE"

Each character is tanslated into one 8-bit byte of storage.

11.1.5.2.1.2 Hexadecimal Constant

A Hexadecimal constant can be any even number of Hex digits,

The hexadecimal constant must be enclosed in single quotation marks. Examples are:

NAME	OPERATION	OPERAND
DATA1	DC	'1FE036'
	DC	'C800'

11.1.5.2.1.3 Decimal Constants

A decimal constant can be any number less than 256.

OPERATION	OPERAND
DC	30
DC	-9

The decimal constant is converted to an 8-bit integer.

11.1.5.2.1.4 Address Constant

An address constant is a 2 byte storage address that is translated into a constant.

OPERATION	OPERAND
DC	LOOP+2
DC	TABLE

The following examples show how a single DC instruction can be used to define different types of data. Operands in the DC statement may be strung together, separated by commas, as long as they fit on a single source statement.

LOCATION	DATA	LABEL	OPERATION	OPERAND
0000	0F00	ADDR	DC	'0F00', "CAT", 23,
0002	4341			
0004	54			
0005	17			
0006	0006		DC	*, ADDR+2, @ADDR
0008	0002			
000A	8000			

11.1.5.2.2 Define Storage DS

NAME	OPERATION	OPERAND
A symbol (Optional)	DS	An expression

The DS assembler instruction is used to reserve storage areas. The value of the expression in the operand entry determines the number of bytes reserved. The label of a DS statement is the location of the first byte reserved. No data is generated and the storage area reserved is not preset to any value.

EXAMPLE:

NAME	OPERATION	OPERAND
INAREA	DS	80
OUTPUT	DS	'0100'

11.1.5.3 Assembler Control Instructions

Assembler Control Instructions are used to control the location counter. None of these instructions generate machine instructions or constants in the object program.

11.1.5.3.1 ORG - Set Location Counter

NAME	OPERATION	OPERAND
Not used	ORG	expression

The ORG PSEUDO-OP is used to control the location counter. The ORG causes the location counter to be set to the value of the expression in the operand entry.

The location counter is initialized to zero before each assembly. If no ORG assembler instruction appears at the beginning of the program, the location counter will begin at zero.

Symbols appearing in the operand of the ORG must be previously defined.

EXAMPLES:

```
ORG     '0100'
ORG     20
ORG     ADDR+3
```

11.1.5.3.2 END - End Assembly

OPERATION	OPERAND
END	An Expression optional

The END assembler instruction terminates the assembly of the program. The value of the expression, if present, designates the place in the program where control is transferred after the program has been loaded. If an expression is not present, no automatic transfer of control takes place after loading.

An example follows:

```
NAME        OPERATION       OPERAND
            ORG             100
PLACE 1     LI              DATA 2
              .
              .
              .
            END             PLACE 1
```

11.1.5.4 Summary of Assembler Instructions

SYMBOL DEFINITION INSTRUCTIONS

- DC Define Constant, used to specify the following data types.
    1. Character Constant
    2. Hexadecimal Constant
    3. Address Constant or
    4. Decimal Constant
- DS Define Storage
- END End Assembly
- EQU Equate A Symbol
- ORG Set Location Counter

11.1.6 Input Format

The source program may be prepared as a deck of cards or as a source tape. If cards are to be the source input, they will be prepared with one instruction per card. The following format is recommended, although the assembler will accept free format.

1. Columns 1 through 6 - Label
2. Column 9 - (0.2.8) punch "space" character
3. Columns 10 through 14 - Symbolic Operation Code
4. Column 15 - (0.2.8) punch "space" character
5. Column 16 on - Symbolic Operands
6. Column 35 - (0.2.8) punch "space" character
7. Column 36 on - Comments

Items 1, 6 and 7 are optional. Item 2, 4, and 6 are optional depending on the type of card reader used. If the card reader used generates a column strobe, the (0.2.8) "space" character may be replaced by blank columns-keypunch space bar. The cards must be front slashed (IBM form X28-6509-2), and prepared on an IBM 029 or 026 keypunch. If the operands occupy more space than that suggested above, the comment field may be moved right. The comment field is restricted in length because of the narrow pages that a teletype types. Longer statements will be truncated to 71 listing spaces total.

The Assembler needs to see only one space between source statement fields, superfluous blank columns (spaces) are ignored. The Assembler reformats the input statements which may cause the appearance of the listing to differ from the appearance of the source cards.

Source tapes are prepared by the Text Editor for input into the assembler. Each field of the source tape is separated by one or more blank characters. The code is ASCII. Each instruction is terminated by a carriage return. If the teletype is the source input device, each instruction on the source tape will be separated from the next by a minimum of six delete characters. This is necessary because of the start and stop characteristics of the teletype paper tape reader. This format is normally produced by the Editor and is of no real concern to the user.

11.1.7 Operating Instructions for the Model 1 Assembler

11.1.7.1 General Description

The Assembler will accept source statements from a card reader, teletype, high speed paper tape reader, or cassette tape.

11.1.7.2 Configuration

The Assembler will run on any Model 1 Processor with 4K or more of core memory. The minimum device configuration is a teletype.

11.1.7.3 Tape Format

The Assembler is provided in standard loader format. Refer to the Loader Descriptions (11.2) for an explanation of the tape organization and loading sequence.

11.1.7.4 Loading Procedures

The Assembler tape is loaded with the Model 1 General Loader. Prior to loading, the Binary Input Device specification at X '014A' must be set to select the desired loading device.

The following steps are required to load the assembler.

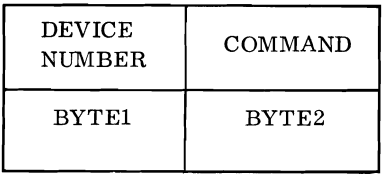
1. Load the General Loader into the top available page of memory using the Incore Loader. The processor will halt.
2. Place the assembler tape in the tape reader with leader over the read station.
3. Enter RUN mode to begin loading the assembler.
4. When all of the program has been loaded, the tape will stop, and control is transferred directly to the assembler at '180' for PASS1. The processor then halts.

11.1.7.5 Device Selection

The Assembler uses three 2-byte locations in the Device Definition Table as follows:

LOCATION	NAME	USED FOR
'014C' - '014D'	BOUTDV	Selection of the punch device
'014E' - '014F'	SINDV	Selection of the source input device
'0150' - '0151'	SOUTDV	Selection of the list device

These Locations must be set up prior to starting the assembler. Information contained in these halfwords are of the form:



The appropriate halfwords for various devices are shown below:

Teletype input/output via serial port	0000
Teletype input	0294
Teletype output	0298
Line Printer output	6280
High Speed paper tape input	0399
High Speed paper tape output	0392
Card Reader input	04A0

Various configurations are as follows:

```
'014C'  '0000'  Teletype punch device (serial port)
'014E'  '0000'  Teletype source input device (serial port)
'0150'  '0000'  Teletype list device (serial port)

'014C'  '0298'  Teletype punch device
'014E'  '0294'  Teletype source input device
'0150'  '0298'  Teletype list device

'014C'  '039A'  High Speed punch device
'014E'  '0395'  High Speed papertape source input device
'0150'  '0298'  Teletype list device

'014C'  '039A'  Speed High punch device
'014E'  '04A0'  Card reader source input device
'0150'  '6280'  Line printer list device
```

11.1.7.6 Operating Procedures

After loading the Assembler tape, control is transferred directly to the Assembler. The Assembler then halts to allow device preparation. Enter RUN mode to proceed with the assembly.

If the Assembler needs to be restarted, use the following procedure:

1. Set the Data/Address switches to 0180
2. Select Address mode and depress EXECUTE
3. Select RUN mode and depress EXECUTE. The Assembler will perform some initialization then halt.

At this point, the operator should place the source tape or card deck in the source input device, adjust the list device to top of form then depress EXECUTE.

After reading the source program, the Assembler prints the symbol table on the List device than halts.

Replace the source tape or card deck in the source input device, prepare the punch device and enter RUN mode to start pass 2. The Assembler generates the assembly listing and object tape on the second pass. If for any reason, pass two must be restarted, set the Data/Address switches to '0182' and repeat step 2 and 3 above.

11.1.7.7 Symbol Table Size

The symbol table begins after the last location in the Assembler program. The top of the sy mbol table is defined by the page address contained in location 'F0'.

At load time the value is '10' which defines a 4K core memory. If more symbol table space is required change the entry in location F0' as shown below.

CORE AVAILABLE	ENTRY IN LOC 'F0'
4K	'10'
6K	'18'
8K	'20'
10K	'28'
12K	'30'
14K	'38'
16K	'40'

The maximum number of symbols that can be defined depends on the length of the symbol. The table space required in bytes equals the number of characters in the symbol plus two.

The symbol capacity is approximately 220 symbols for a 4K machine.

11.1.7.8 NO-PRINT and NO-PUNCH Options

Assemblies may be performed without printing of the listing or punching of an object tape if so desired by the user to save assembly time.

If no listing is desired, zero location 'OOEE' before the assembly is begun

If no object tape is desired, zero location 'OOEF' before the assembly is begun

To restore the print or punch option, write any non-zero value into the above locations.

11.2 MODEL 1 IN-CORE LOADER

11.2.1 Introduction

The Model One In-Core Loader comprises the basic 8-bit loader and Device Definition table which reside in page one. A portion of this area of core memory is usually reserved for this sequence.

The In-Core Loader must be manually entered into memory when the processor is first turned on.

The In-Core Loader serves two basic functions:

1) The 8-bit loader will read 256 bytes (one page) of 8-bit binary input data into a specified page of memory and then branch to the first address, address '00', on that page. The Model One General Loader, program number 08-005, or any 256 byte binary object tape constructed by the user may be loaded in this manner.

2) The Device Definition Table provides a limited degree of device independence by specifying which devices are to be used by standard programs.

There are two versions of the In-Core loader. The first, shown in Table 11-5, is used when the binary input device is a teletype interfaced through the Serial I/O port. The second version, shown in Table 11-6, is used when the binary input device (Teletype, High Speed Paper Tape reader, or Cassette Tape) is interfaced to the multiplexor I/O bus.

11.2.2 Loader Description

The 8-bit loader stores 8-bit bytes into consecutive byte locations beginning at address '00' of the destination page and ending at address 'FF' of the destination page. The destination page is specified by the contents of location '0152' which the user may adjust to any available page other than page one. Page one ('01') may not be specified as this would allow the loader to be over-written. Specifying a non-existent page will cause the tape to be read, but nothing will be loaded.

The In-Core loader does not skip leader so it is important that the first data character on the tape be positioned over the read station. Also, when loading from the teletype, tape motion must be manually started. After the loader is started at '0140', with an ASR33, toggle the reader switch to START. With an ASR35, put the reader switch in RUN with the teletype Mode switch in the KT position.

TABLE 11-5
IN-CORE LOADER, SERIAL I/O VERSION

```
0140  7154  LOAD     BAL   GETBYT
0142  CB52           ST    PAGE,I
0144  D553           ISZ   PAGE+1
0146  6140           B     LOAD
0148  6352           B     PAGE,I
                     *
                     * DEVICE DEFINITION TABLE
                     *
014A  0294  BINDV    DC    '0000'
014C  0298  BOUTDV   DC    '0000'
014E  0294  SINDV    DC    '0000'
0150  0298  LISTDV   DC    '0000'
                     *
0152  0000  PAGE     DC    '0000'
                     *
0154  0000  GETBYT   DC    '0000'
0156  8210           C     '10',NC
0158  6156           B     *-2          WAIT FOR START
015A  90F7           LI    -9
015C  C97B           ST    CHAR
015E  28             AA
015F  C97A  COM      ST    COUNT        TIMER
0161  9009           LI    9            13.5 or 9 MS
0163  10             AO
0164  6563           B     *-1,NZ
0166  D57A           ISZ   COUNT
0168  6161           B     COM+2
016A  9000  TIME     LI    0
016C  CD7B           ISN   CHAR
016E  6354           B     GETBYT,I     END
0170  8010           C     '10'
0172  8856           SH    1,CI         SHIFT IN BIT
0174  C96B           ST    TIME+1
0176  90F4           LI    -12
0178  615F           B     COM
017A  00     COUNT    DC    0
017B  00     CHAR     DC    0
```

TABLE 11-6
IN-CORE LOADER, MULTIPLEXED I/O VERSION

```
0140    7154    LOAD        BAL     GETBYT
0142    CB52                ST      PAGE,I
0144    D553                ISZ     PAGE+1
0146    615C                B       NEXT
0148    6352                B       PAGE,I
                *
                * DEVICE DEFINITION TABLE
                *
014A    0294    BINDV       DC      '0294'
014C    0298    BOUTDV      DC      '0298'
014E    0294    SINDV       DC      '0294'
0150    0298    LISTDV      DC      '0298'
                *
0152    0000    PAGE        DC      '0000'
                *
0154    0000    GETBYT      DC      '0000'
0156    D14A                L       BINDV
0158    00                  ADR
0159    D14B                L       BINDV+1
015B    01                  OC
015C    07      NEXT        RDS
015D    615C                B       *-1
015F    6354                B       GETBYT,I
```

11.2.3 Device Definition Table

The Device Definition Table contains four halfwords. Each halfword specifies one device. The format is:

DEVICE NUMBER	OUTPUT COMMAND

The left byte specifies the device number, the right byte contains the output command required to start that device. The four halfwords are used as follows:

X'014A' BINDV Binary Input Device

Used by loaders to select the load device.

X'014C' BOUTDV Binary Output Device

Used by the Assembler, Text Editor, Unloader etc. To select the punch device.

X'014E' SINDV Source Input Device

Used by the Assembler, and the Text Editor to select the source Input Device.

X'0150' LISTDV List Device

Used by the Assembler to select the list device.

The range of devices that can be used varies with different programs. The In-Core loader only references BINDV at '014A'. Table 11-6 shows the Device Table entries for a teletype. Table 11-7 shows device table entries for other devices.

TABLE 11-7
DEVICE DEFINITION TABLE ENTRIES

DEVICE	BINDV	BOUTDV	SINDV	LISTDV
TTY (Serial)	'0000'	'0000'	'0000'	'0000'
TTY (Parallel)	'0294'	'0298'	'0294'	'0298'
HIGH SPEED PAPER TAPE	'0395'	'039A'	'0395'	'039A'
CASSETTE TAPE	'4581'	'4582'	'4581'	'4582'
CARD READER	-	-	'04A0'	-
LINE PRINTER	-	-	-	'6280'

11.3 MODEL 1 GENERAL LOADER

11.3.1 Introduction

The Model One General Loader is a relocatable program designed to load standard format binary object tapes as generated by the Model One Assembler.

The Input Device for loading is specified by the Binary Input Device entry in the Device Definition Table. See In-Core Loader description.

11.3.2 Loader Features

When reading binary data from tape, blank tape and illegal characters are skipped and checksums are checked after each binary record is read. After loading the last record, the loader will automatically transfer to the program loaded if specified on the object tape.

11.3.3 Standard Loader Format

Standard format binary object tapes are divided into variable length records containing 128 bytes, or less, of object data. Records are separated by blank characters. Each character (frame) punched on paper tape represents one hexadecimal digit of information. Therefore, two frames of paper tape are required to represent one byte of object data. This representation is known as zoned ASCII. The first four bits in a frame are the ASR33 zones and the last four bits are the actual data.

The zones have been selected to produce a non-printing set of ASCII characters, avoiding the characters XON, XOFF, TON, TOFF and WRU that could interfere with teletype usage. Table 11-8 shows the character set recognized. All other characters are ignored by the loader.

TABLE 11-8
VALID LOADER CHARACTERS

Zone	Data	Hex Data
1001	0000	0
1000	0001	1
1000	0010	2
1000	0011	3
1000	0100	4
1001	0101	5
1001	0110	6
1001	0111	7
1001	1000	8
1001	1001	9
1001	1010	A
1001	1011	B
1001	1100	C
1001	1101	D
1001	1110	E
1001	1111	F

The first data byte (two frames) of a record is an 8-bit exclusive-or checksum of every remaining character in the record. The next two

bytes (four frames) are the origin address for the variable length object data that follows immediately. The tape therefore appears as shown on Figure 11-1.

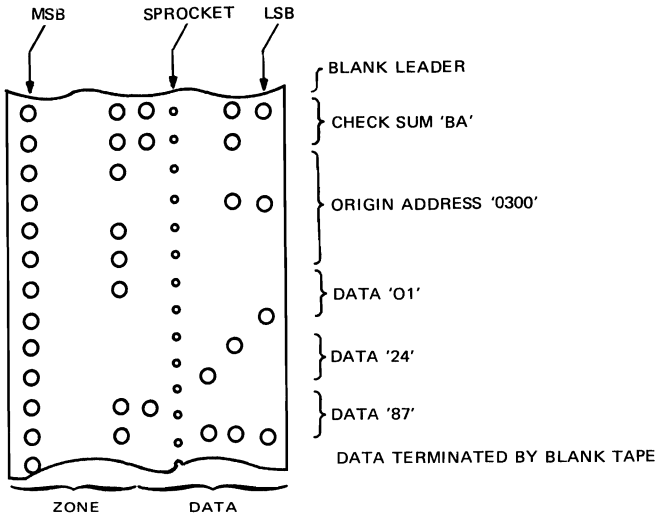


Figure 11-1. Tape Format

Following the last data record on the binary object tape is a special end record containing only a check sum and an address. This record differs from data records in that bit 0 of the address is set. If bit 1 is also set (remember that Model One Memory addresses are only 14 bits) the loader halts. If bit 1 is reset, the loader transfers control to the user program at the address specified. See Figure 11-2.

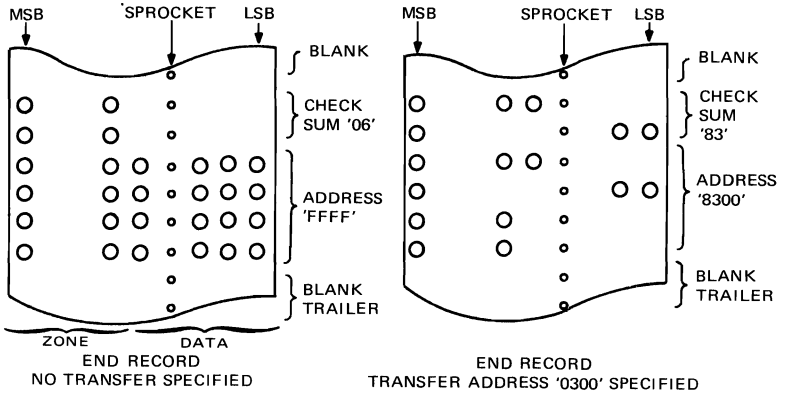


Figure 11-2. End Record

11.3.4 Loader Tape Format

The loader tape is provided in 8-bit format and may be loaded into any page of memory other than page one by the In-Core loader. The General Loader occupies the Upper 128 bytes of the page. After loading, the General Loader halts at address '82' of the page, ready to load a user object tape.

11.3.5 Operating Procedures

The Steps required to load and operate the General Loader are summarized below:

1. Manually load the In-Core loader and adjust the Binary Input Device definition. NOTE: The General Loader uses the input driver of the In-Core loader.
2. Adjust the destination page specification in location '0152' to the page where the loader is to reside.
3. Place the loader tape in the tape reader with the first non-zero character over the read fingers or photo diodes.
4. Enter '0140' in the display switches, select ADR Mode and depress EXECUTE.
5. Select RUN Mode and depress EXECUTE.
6. If a teletype is the input device, tape motion must be manually started.
7. The entire tape is read and the Program will halt with 'XX 82' in the Location Counter where XX is the page and 82 is the starting address of the General Loader.
8. Put the object tape to be loaded in the tape reader with leader over the read fingers or photo diodes and enter RUN Mode.
9. If check sum errors are detected during tape input, the tape will stop and the processor will halt at address 'D4' on the loader page. When this occurs, re-position the tape to the previous record gap, and depress EXECUTE to re-read the record.
10. When the load is complete, and if no end transfer address is specified in the end record, the Processor will halt at address '82' on the loader page, ready to load another tape. If an end transfer address is specified, the loader will transfer directly to the location specified.
11. If more tapes are to be loaded, repeat from step 8.
12. To re-enter the loader at any time, enter the RUN Mode at location '82' of the loader's resident page.

11.4 THE MODEL 1 UNLOADER

The Mod 1 Unloader is a relocatable 1 page program used to output from core to tape in standard loader format. It is especially useful to the 2K user who cannot use the assembler. He has the capability of writing a program in machine language, keying it into the Processor manually with DBUG or the memory write function, and then punching a loader format object tape. The Unloader punches core between 2 limits specified by the user on a specified device. The program is available in binary format and is loaded by the Incore loader. Instructions for use are as follows:

1) Select a page for the Unloader program to reside, and enter the page number into location '0152' of the In-Core Loader.

2) Set desired Binary Output Device and Command in location '014C - 014D' of the Device table.

3) Load the Unloader program tape with the In-core loader. The processor will halt.

4) Using the memory write function of the console, place the low limit (14 bit address) desired by the user into locations 'FC' - 'FD' of the Unloader's resident page.

5) Place the high limit address in locations 'FE'-'FF' of the Unloader's resident page

6) Select RUN Mode and depress EXECUTE. The Unloader will punch core from the low limit to the high limit on the specified device.

7) When finished the Processor will halt and 4, 5 and 6 may be repeated.

11.5 HEXADECIMAL DEBUG PROGRAM DESCRIPTION (DBUG)

11.5.1 Introduction

DBUG is an on-line, real-time relocatable hexadecimal debug program. Its purpose is to provide maximum assistance in debugging a user program while using a minimum of memory storage. The user of DBUG directs the debugging operation by entering directives and associated data via the teletype keyboard. The response to these inputs is shown on the teletype page printer. Table 11-9 provides an Index of Directives.

11.5.2 Terminology

The term cell, as used in the following discussion, refers to a 2-byte (sixteen-bit) memory location, defined by the address of the left-most byte.

An open cell is the cell that is currently available for modification. To operate on a cell in memory, it must be made the current open cell. Only one cell at a time is considered open.

TABLE 11-9
INDEX OF DIRECTIVES

1. Cell Examination and Modification		2. Program Control	
↓	OPEN CELL	X	INSERT BREAKPOINT
LINE FEED	OPEN NEXT SEQUENTIAL CELL	Z	ZAP A BREAKPOINT
CARRIAGE RETURN	OPEN PRECEDING CELL	K	KILL ALL BREAK-POINTS
R	OPEN A-REGISTER	G	GO EXECUTE
•	MODIFY CONTENTS	W	GO WAIT
T	TRANSFER TO CONTENTS		

Directives are instructions to DBUG and consist of a single character, other than 0-9 and A-F. Directives are given to DBUG through the teletype keyboard. See Figure 11-3. Each directive initiates an action. Some directives are to be preceded by an argument.

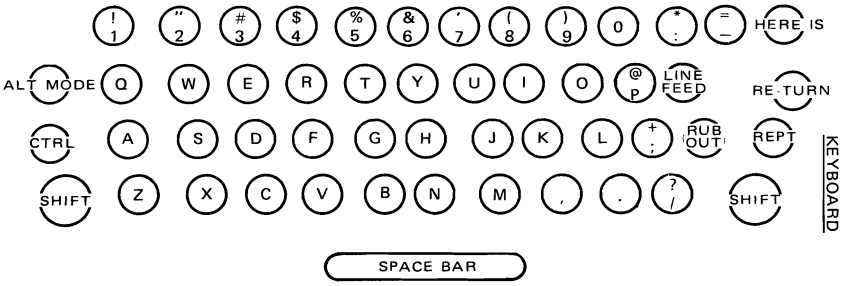


Figure 11-3. ASR 33 Teletype Keyboard Layout

The argument is the address or data input which will be used by the directive. Leading zeros are not necessary. Data inputs are in a hexadecimal format using the characters 0-9 and A-F. All other characters are assumed to be directives to DBUG. An undefined directive given to DBUG is ignored and no action is taken.

11.5.3 Configuration

DBUG requires a teletype keyboard for the input of directives and the teletype page printer for the output of responses.

11.5.4 Tape Format

DBUG is provided in standard loader format. Refer to 11.9.3

11.5.5 Features Available in DBUG

Memory cell examination. Memory cell modification. Multiple breakpoints. Execution of User's Program. Halt in User's program. Accumulator and Carry Bit examination and modification.

11.5.6 Description of Operations

The directive for each operation is defined by showing the directive teletype key encircled to the left along side its title, and followed by a description of its function.

The underline is used in the examples to differentiate between the user's input and DBUG's output response. The user's input is underlined.

11.5.6.1 Cell Examination and Modification

The directives described in this section provide memory cell examination and modification. These directives are generally preceded by some hexadecimal input. Hexadecimal inputs are accepted until a directive is received; the last four hexadecimal characters are then used as the address or data. The input of leading zeros is not necessary.

(Ø) "OPEN CELL"

The space bar is the cell examination directive. Typed after an address, it causes that cell's address, and the content of that cell is to be printed. If no address was specified prior to typing the space bar, the address of cell zero and its contents will be printed. For example, if the user types 1FEØ, DBUG will output a carriage return, line feed, the address 01FE and the content of the reference cell XXXX.

Example: 1FEØ
01FE XXXX

(LINE FEED) "OPEN NEXT SEQUENTIAL CELL"

To open the next cell in sequence, the user types a line feed. The current open cell is closed, and the address and content of the next sequential cell are printed.

Example:

```
1FEØ
01FE   XXXX   LINEFEED
0200   NNNN
```

(RETURN) "OPEN PRECEDING CELL"

To open the previous cell instead of the next cell in sequence as in LINE FEED above, the user types a carriage return. The current open cell is closed and the address and content of the previous cell are printed.

Example:

```
1FEØ
01FE   XXXX   RETURN
01FC   NNNN
```

(T) "TRANSFER TO CONTENTS"

This directive causes the content of the current open cell to become the address of the new open cell. The address and content of the new open cell are printed.

Example:

```
1FEØ
01FE   03E0   T
03E0   XXXX
```

(•) "MODIFY CONTENTS"

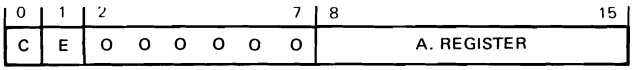
The period is the directive to modify the open cell. The content of the current open cell is replaced by the four hexadecimal characters entered just prior to the period. The input of leading zeros is not necessary, although acceptable as in the following example. If no hexadecimal entry is made, the cell is zeroed.

Example:

```
1FEØ
01FE   03F0   045C.
01FE   045C   .
01FE   0000
```

(R) "OPEN A-REGISTER"

The status of the A Register, the carry flag and the Interrupt Enable Flag at the time of entry to DBUG, or upon encountering a breakpoint, is saved in a reserved cell within the DBUG program. The format of this cell is:



This cell may be accessed for examination or modification by the R directive. This cell becomes the current open cell and may be operated as such.

Example: (R)

```
AREG    X0XX    7.
AREG    0007    8005.
AREG    8005
```

The A-Register, Carry flag and Interrupt Enable flag are restored from this cell when the G or W directive is executed. If modified, they assume their new values upon execution of the G or W directive.

11.5.6.2 Program Control

The program control directives (X, Z, K, G, and W) permit the user to direct the operation of this program through the use of DBUG. At a point within the user's program, it is frequently desirable to return control to DBUG. This is accomplished by inserting a breakpoint at the address at which the transfer of control is to take place. When a breakpoint is encountered during the execution of a program, control returns to DBUG and the message BKPT is printed. Along with the address and contents of the breakpointed location. This becomes the current open cell.

(X) "INSERT BREAKPOINT"

The X directive causes a breakpoint to be inserted at the address specified by the preceding hexadecimal input. If no address is specified, the breakpoint will be inserted in the current open cell.

Up to eight breakpoints at a time may be resident in the user's program. If a breakpoint insertion is requested and eight breakpoints have already been inserted, the message BPOF (Breakpoint Overflow) is printed and the address of the current open cell or the specified cell and its content is printed.

When a breakpoint is inserted, the address at which the breakpoint is inserted and its new contents are printed. That address becomes the current open cell.

Example: The breakpoint is inserted on the currently open cell:

```
1FEB
01FE    03E0    X
01FE    72FE
```

An address is specified:

```
1FEB
01FE    03E0    25AX
025A    72FE
```

Following are some restrictions to observe pertaining to breakpoints.

1. Regardless of the size of the instruction word at the referenced cell, two successive bytes are replaced by the breakpoint value '72FE'. If the breakpoint is set on a one-byte instruction no attempt should be made to execute the next instruction before the breakpoint is removed.
2. Core locations '00FE' and '00FF' are used by DBUG to process breakpoints, so the contents of these locations are destroyed if the X directive is used.
3. The user should be careful not to set breakpoints which overlap (are within one byte of one another).

Two bytes of a user program are needed to set a breakpoint. A BAL indirect instruction to address "FE" of page zero (HEX '72FE') is inserted. Thus, the user will overlay two (2) bytes when he sets a breakpoint.

(Z) "ZAP BREAKPOINT"

The Z directive causes the removal of a single breakpoint. If the Z directive is preceded by a hexadecimal input, the breakpoint at the address specified by that input is removed. If no address was specified, the breakpoint is removed from the current open cell. The breakpoint is removed and the address and restored content of the cell are printed. If no breakpoint is found at the address indicated, the message NOBP and the address and content of the cell are printed.

```
Example:   01FE   72FE   Z
           01FE   03E0   3E0Z
           03E0   XXXX   NOBP
```

(K) "KILL ALL BREAKPOINTS"

The K directive removes all resident breakpoints from the user's program.

(G) "GO"

The G directive causes DBUG to restore the A Register, Carry flag and interrupt enable flags and transfer processor control to the user's program. If the G directive is preceeded by a hexadecimal input, execution will begin at the address specified by that input. If no address is specified, execution begins at the current open cell.

A carriage return and line feed are output in response to this directive to indicate that transfer of control is about to take place.

Example: 0200G

(Carriage return, line feed, control transferred to 0200.)

When DBUG is loaded, the A Register, the carry flag and Interrupt Enable flag are preset to 0. Therefore unless these are modified by DBUG, the G directive will use these preset values.

Ⓦ "GO WAIT"

The W directive causes DBUG to restore the A Register, Carry Flag and Interrupt Enable Flags and place the Processor in the WAIT State. If the W directive is preceeded by hexadecimal input, the Processor is halted at the location specified by that input. If no address is specified, the Processor is halted at the current open cell.

A carriage return and line feed is output in response to this directive to indicate that transfer of control is about to take place.

Example: 200W

(carriage return, line feed)

Processor is halted at location 0200.

The directive facilitates single step console debugging of a user program. Select RUN mode or HALT Mode and depress execute to cause the instruction at the referenced cell to be executed.

11.5.7 Loading Procedures

DBUG is a relocatable program in absolute form. The tape provided to the user is in standard loader format and must be loaded with the Mod 1 General Loader. It is loaded as an absolute program into pages 4-7. It then refers to a 1-BYTE pointer in memory, and relocates itself to a 4 page block of core memory specified by that pointer. The pointer byte is location '0400'.

DBUG may relocate itself to any 4 page block of core memory attached to the processor except those in the lower 2K. After relocation DBUG takes control and is ready to receive directives. The pointer byte can be set to any number between X'08' and X'3C' representing the first of 4 consecutive pages of core memory in which DBUG will reside. If the pointer contains a value other than '08' - '3C' then DBUG will remain in pages '04' - '07'.

The following is a description of the loading procedures:

1. Set loc '0400' to the desired core pointer value.
2. Using the In-core loader, load the Model 1 General Loader into any page other than 1 or 4-7 as described in Section 11.2.

3. Enter RUN Mode to load the DBUG program with the Model 1 General Loader. DBUG will relocate and assume control.

ENTRY POINTS: To enter DBUG for another program, a processing loop, or any external situation, EXECUTE at location '00' of the top page of the 4 page block occupied by DBUG.

Example: If DBUG was relocated to page '08' then the entry address is '0B00'

CORE USAGE: Not all of the lower page of the 4 pages is required by DBUG.

After relocation, if the TTY used by DBUG is serial I/O driven, the user may overwrite '00' - 'BF' of the lower page. If the TTY is not serial driven, the user may overwrite the entire lower page.

11.6 MODEL 1 TEXT EDITOR PROGRAM DESCRIPTION

11.6.1 Introduction

The Model One text editor, Program Number 08-007, is an on-line, interactive text editing program. It is designed to create and modify character-oriented text material which is stored on paper tape or input through the teletype keyboard. The text may be an assembly language program or any text in the literal sense.

The editing process is directed by an operator through the keyboard of a teletype terminal. Upon receiving a keyboard input directive, the editor will read text from a specified input device into a designated area of core memory. The user can examine, delete and/or modify the text while it remains in core memory. When the editor receives a keyboard output directive, the revised text can then be output to a specified output device.

11.6.2 Program Structures

11.6.2.1 Operating Modes

The editor has two modes of operation: Command and Edit. The program indicates the current mode by printing, in column one on the teletype, a left arrow (←) for the command mode or an asterisk (*) for edit mode. In the Command mode, the program accepts keyboard commands which specify an editing procedure or which specify a text input or output operation.

From an Edit command, the program enters the edit mode. Edit mode allows the user to insert or append text after which control returns to the command mode.

11.6.2.2 Basic Unit

The basic unit of stored text is a variable length line from 1 to 67 ASCII characters long including the line terminating carriage return (CR). Each line of input is stored in the text buffer as it is received. If the text buffer contains a symbolic source program, each source statement is one line of text.

Lines of text in the text buffer have unique decimal addresses which are sequenced in ascending order; the first line in the buffer has address number one (0001). This allows editing of any line by line address rather than core location address.

11.6.2.3 Line Addressing

A specific line can be referenced by its decimal number address. To examine line n, type the decimal number followed by a carriage return. The teleprinter will list:

n ZZZ....Z

Where n is the four digit line number, and Z the text contained in line n. This becomes the line currently available for modification and is called the open line. To examine the last line in the text buffer, type the letter Q. The teleprinter will list:

n ZZZ.....Z

Where n is the four digit line number, and Z the text contained in the last line in the text buffer.

Any attempt to examine a non-existent line will result in an error message. See Section 11.6.2.7. The execution of some editor commands will change the position of a line in the text buffer, and consequently change the line number. This number change does not affect the contents of the line. See Section 11.6.2.6 for Command examples.

11.6.2.4 Command Formats

Commands are entered through the keyboard in one of the following formats:

FORMAT	DESCRIPTION
X	Editor performs Command X
nX	Editor performs Command X on n lines

Where n is a 4 digit maximum decimal number and X is a command specifying the operation to be performed.

Editor commands are described in Table 1.

If the command is illegal (Section 11.6.2.7), no action is taken and the editor returns to the Command mode (←).

11.6.2.5 Commands

The three main functions of the editor are input, modification, and output. The input commands are used to enter text into the text buffer. The modify command manipulates the text stored in the text buffer. The output commands produce a hard copy of the text on a specified output device. Table 11-10 contains the definitions for the command repertoire. Section 11.6.3.2 explains input/output device specifications.

TABLE 11-10
COMMAND REPERTOIRE

Function	Keyboard Input Command	Response	Definition	Description
Input	A	*	Append	The editor enters the edit mode and accepts input from the Keyboard. The typed text line is appended following the last line (if any) in the text buffer. Each line of input is terminated with a CR. After an * response, the Append operation is terminated by typing ↑. After termination, the last line input, now the open line, and its decimal number address are printed. The program returns to the command mode (←)
Input	nA	None	Append n Lines	The editor enters the edit mode after which n lines are read from the source input device.[1] The read operation may be aborted by depressing

TABLE 11-10
COMMAND REPERTOIRE (Continued)

Function	Keyboard Input Command	Response	Definition	Description
				console switch 15. After manual or normal termination, this command continues as the A command.
Input	I	*	Insert	The editor enters the edit mode and accepts text lines from the keyboard to be inserted preceding the open line. Insertions are made in the order in which lines are input. Each line is terminated with a CR. After an * response the Insert operation in terminate by typing ↑ or by depressing console switch 15. Upon termination, the open line with its corrected decimal address will be printed. Control returns to the command mode.
Input	nI	None	Insert n lines	The editor enters the edit mode after which n lines are read from the source input device[1] and are inserted into the text buffer as described for the I command. The read operation may be aborted by depressing console switch 15. After manual or normal termination this command continues as the I command.

TABLE 11-10
COMMAND REPERTOIRE (Continued)

Function	Keyboard Input Command	Response	Definition	Description
Modify	D	None	Delete	The editor deletes the current line. The line following the current open line is now the open line and will be printed along with its corrected line number. If the last line in the buffer was the open line and it is deleted, the new last line is now the open line and it is listed Control returns to the command mode (←)
Modify	nD	None	Delete n lines	The editor deletes n lines of text beginning with the current open line. The line following the last line deleted or the last line in the text buffer, if less than n lines remained, becomes the open line and it is listed. Control returns to the command mode (←).
Output	P	None	Print	The line number and content of the current open line are printed on the Teletype. Control returns to the command mode (←).
Output	nP	None	Print n lines	The editor prints n lines beginning with the current open line. If less than n lines remain, output terminates after printing the last line in the text buffer. Output may be aborted by depressing console switch

TABLE 11-10
COMMAND REPERTOIRE (Continued)

Function	Keyboard Input Command	Response	Definition	Description
				15. After manual or normal termination control returns to the command mode (←).
Output	O	None	Output punched tape	The entire text buffer is punched in the standard source tape format[2] on the binary output device[1]. Output may be aborted by depressing console switch 15. After manual or normal termination control returns to the command mode (←).
Set Up	K	←	Kill the text buffer	This command erases the text buffer and returns control to the command mode (←).
Other	nS	None	Skip n lines	The Source Input Device[1] is advanced the number of lines specified. No information enters the text buffer. Skipping may be aborted by depressing console switch 15. After manual or normal termination control returns to the Command mode.

TELETYPE EDITOR CONTROL

KEY	DEFINITION
↑	Return to Command Mode the open line is printed
←	Delete last character typed
RO	Delete line just typed
CR	End of Current Line

Footnotes:

1. See Section 11.6.3.2, I/O Device Selection
2. See Section 11.6.3.4, Tape Format

Note: The symbol CR refers to a carriage return. Underlined characters are processor responses.

11.6.2.6 Command Examples

See Table 11-10 for command definitions

a. Append

```
←        A
*        APPEND LINES TO (CR)
*        THE TEXT BUFFER (CR)
*        TERMINATE INPUT BY (CR)
*        TYPING AN UP-ARROW
*        ↑
0004     TYPING AN UP-ARROW
←
```

6. Open last line

```
← Q
0004     TYPING AN UP-ARROW
←
```

c. Open line

```
← 3 (CR)
0003     TERMINATE INPUT BY
←
```

d. Insert

← I

* INSERT LINES (CR)

* INTO THE TEXT BUFFER (CR)

* ↑

0006 TERMINATE INPUT BY

←

e. Delete

← D

0005 TYPING AN UP-ARROW

f. Print

← 1 (CR)

0001 APPEND LINES TO

← 5P (CR)

0001 APPEND LINES TO

0002 THE TEXT BUFFER

0003 INSERT LINES

0004 INTO THE TEXT BUFFER

0005 TYPING AN UP-ARROW

11.6.2.7 Errors

The error message for an improper editor command entry or for a line of text (from any input device) which exceeds the character limit, is the question mark (?). If a command entry error is made, no action is taken upon the information in the text buffer. The program responds with the error message (?) and remains in the command mode (←). If the text line exceeds 67 characters, the error message (?) will be printed and control is transferred to the command mode (←). None of the characters in the line will be entered into the text buffer. If a typing error occurs and is discovered before typing the CR, the mis-

take may be corrected. Corrections are made by typing a left arrow (←) which deletes the last character input, or by typing a Rubout (RO) which deletes the entire line. Control remains in the edit mode.

Another error flag is the exclamation point (!) which means the text buffer has overflowed. When this happens, the line which caused the overflow is not entered into the text buffer. The program returns to the command mode.(←) To enter more information, it is necessary to delete one or more lines from the buffer or adjust the pointer to the top of the text buffer. See Section 11.6.3.5.

11.6.3 Operating Procedures

11.6.3.1 Loading

The Model 1 text editor, program number 08-007, requires '2K' bytes of memory including a 1000 byte text buffer. To load the editor, use the Model 1 General Loader, program number 08-005. Refer to Section 11.3 for use of the loader.

11.6.3.2 I/O Device Selection

Prior to executing the editor the appropriate 2 byte blocks in the Device Definition Table (11.2.3) should be set up in the following format:

Device Number	Output Command

The appropriate halfwords for various devices is shown below.

'0000'	TTY I/O via serial I/O Port
'0294'	TTY tape reader input (no printing)
'02A4'	TTY input (with printing)
'0298'	TTY output
'0399'	High speed paper tape input
'0392'	High speed paper tape output

Device selection locations in the Device Definition table appropriate to the editor are

LOCATION	NAME	USE WITH COMMANDS:
'014C'	BOUTDV	Output
'014E'	SINDV	Append, Insert, Skip

11.6.3.3 Starting Location

Starting the Editor at location '0004' causes the text buffer pointers to be reset. The program is initialized and the Command Mode (←) is entered.

Starting the Editor at location '0000' causes the Command Mode (←) to be entered without affecting the Text buffer. Initializing the processor causes execution to restart at location '0000'.

11.6.3.4 Tape Format

Punched tapes produced by the Output (O) command are in the standard source tape format. Each line of text is preceeded by eight rubouts (RO) and terminated with a CR and LF. The format for each character is seven bit ASCII, except the RO which is eight bit ASCII code.

Input tapes to the editor should be in the standard tape format; however, the minimum tape format requirements are that each line must be terminated by a carriage return and contain no more than 67 characters including the carriage return. In addition, successive statements must be separated by at least five or six rubouts, due to the start/stop characteristics of the Teletype reader.

11.6.3.5 Text Buffer Size

When loaded, the editor provides a text buffer for 1000 characters. The user may adjust the size of the Text buffer by inserting the beginning and ending addresses into locations '0008' and '000A' respectively. The text buffer may be located anywhere in core providing it does not overwrite the editor.

TABLE 11-11
EDITOR RESPONSES, CONTROLS, AND ADDRESSES

EDITOR RESPONSES

SYMBOL	DEFINITION
←	Command Mode
*	Edit Mode
?	Error
!	Text Buffer Overflow

TELETYPE EDITOR CONTROL

KEY	DEFINITION
↑	Return to Command Mode the open line is printed
←	Delete last character typed
RO	Delte line just typed
CR	End of Current Line

SPECIAL EDITOR ADDRESS

LOCATION	DEFINITION
'0000'	Restart location. program will not initialize text buffer.
'0004'	Starting location, program will initialize text buffer
'0008'	Location defines first address of text buffer.
'000A'	Location defines last address of text buffer.

Domestic Offices

EASTERN REGION

2 Crescent Place
Oceanport, New Jersey 07757
(201) 229-4040

60 Hickory Drive,
Waltham, Mass. 02154
(617) 899-6287

1800 North Kent St.
Arlington, Va. 22210

CENTRAL REGION

3553 West Peterson Ave.
Chicago, Illinois 60645
(312) 463-9080

1700 Needmore Road
Suite 305
Dayton, Ohio 45414
(513) 277-1142

SOUTHWESTERN REGION

300 North Central Expressway
Richardson, Texas 75080
(214) 238-9656

WESTERN REGION

2390 El Camino Real
Palo Alto, Calif. 94306
(415) 328-0783

8703 La Tijera Blvd.
Los Angeles, Calif. 90045
(213) 670-8386

International Offices

CANADA

Allan Crawford Assoc.
65 Martin Ross Ave.
Downsview, 463 Ontario
(416) 636-4910

157 Saint Charles St. West
Longueuil, Quebec
(514) 670-1212

376 Churchill Avenue
Ottawa, Ontario
(613) 725-3354

721 Aldford Ave.
Annacis Industrial Estate
New Westminster, British Columbia
(604) 524-1161

FAR EAST

Kyokuto Boeki, Kaisha, Ltd.
C.P.O. Box 330
Tokyo, Japan
(270) 7711

UNITED KINGDOM

INTERDATA, Ltd.
Station House
Harrow Road
Wembley, Middlesex
ENGLAND
01-902-3202

INTERDATA®

The Forthcoming Generation