Programmer's Reference Manual

PERIPHERALS

015-000021-00

DATA GENERAL TECHNICAL MANUAL

Programmer's Reference Manual

PERIPHERALS

Ordering No. 015-000021

Printed in the United States of America
Rev. 00, November 1974

NOTICE

Users are cautioned that DGC reserves the right to make changes without notice in the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including, but not limited to typographical, arithmetic, or listing errors.

PREFACE

The Programmer's Reference Manual for Peripherals is designed as a companion volume to the Programmer's Reference Manuals for the NOVA line and ECLIPSE Computers. It furnishes the general principles and the specific details needed to program input/output (I/O) transfers between standard DGC peripherals and all DGC computers.

This manual consists of several sections. Section I introduces the theory of I/O programming on DGC computers and presents several complete examples of I/O programs. The remaining sections of the manual deal with the various families of peripherals supplied by Data General Corporation. Separate chapters in each section provide the specific information necessary to program the individual peripherals in each family. Section II covers terminals; Section III covers hard copy devices. Section IV deals with magnetic tape storage devices, including industry-standard magnetic tape units and the DGC cassette. Section V covers fixed head discs, disc cartridges, and disc packs. Section VI describes analog-to-digital and digital-to-analog conversion with the Analog Data Conversion System. The appendices contain a number of reference tables which supply information about device codes, character codes, and timing figures needed for I/O operations.

Before reading this manual, the programmer should have a basic understanding of the programming of DGC computers, as described in the Programmer's Reference Manual for the NOVA line Computers (DGC 015-000023) and, where appropriate, in the Programmer's Reference Manual for the ECLIPSE Computer (DGC 015-000024). A familiarity with the operation of the DGC assembler, as described in the Assembler Manual (DGC 093-000017), is also recommended, as the programming examples are written in assembly language.

Additional information, of a more technical nature, may be found in the Interface Manual (DGC 015-000031), which describes the technical operation of the I/O bus and explains how to interface a nonstandard peripheral to any of the DGC computers.

This page intentionally left blank

TABLE OF CONTENTS

SECTION I
I/O PROGRAMMING

TABLE OF CONTENTS (Continued)

SECTION I (Continued)

SECTION II
TERMINALS

TABLE OF CONTENTS (Continued)

SECTION II (Continued)

SECTION III
HARD COPY

TABLE OF CONTENTS (Continued)

SECTION III (Continued)

SECTION IV
MAGNETIC TAPES

TABLE OF CONTENTS (Continued)

SECTION IV (Continued)

TABLE OF CONTENTS (Continued)

SECTION V
DISCS

TABLE OF CONTENTS (Continued)

SECTION V (Continued)

TABLE OF CONTENTS (Continued)

SECTION V (Continued)

Page

SECTION VI
ANALOG/DIGITAL
DIGITAL/ANALOG

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

APPENDICES

SECTION I

I/O PROGRAMMING

- OVERVIEW OF INPUT/OUTPUT
- I/O INSTRUCTION SET
- PROGRAM INTERRUPT FACILITY
- DATA CHANNEL FACILITY
- TIMING
- PROGRAMMING EXAMPLES

This page intentionally left blank

OVERVIEW OF INPUT/OUTPUT

INTRODUCTION

Input/output is the process of moving information in a computer system between the central processing unit (CPU) and peripherals such as card readers, line printers, terminals and disc units. Some peripherals, such as card readers, enter information into the system. Some, such as line printers, transfer information out of the system. Some, such as terminals, transfer information in both directions; and others, such as disc units, store information within the system. Peripherals, therefore, can serve two main purposes, they provide the computer with a means of communicating with its surroundings, and they supplement main memory with secondary storage.

A peripheral generally consists of two units, a device and a controller, but it may also include an adapter. The device, sometimes called a drive, a transport or a terminal, is the unit with which information is read, written, stored, or processed. For example, a terminal's keyboard "reads" information; a plotter "writes" information; a magnetic tape transport "stores" information; and an A/D converter "processes" information.

The controller is the interface between the computer and the device, interpreting commands from the computer to the device and passing information between them. For example, a moving-arm disc controller can translate the track address received from the computer into positional commands for the disc drives access mechanism. Once the access mechanism positions the read/write heads, the controller translates the data words it receives from the computer into the sequence of bits required by the disc drive.

The adapter is an additional unit required by some peripherals to complete the communications link between the device and the controller. It performs functions which are similar to those performed in either the device, the controller, or both. Since the adapter cannot be accessed by the programmer, it is usually transparent.

The communications channel through which all information passes between the computer and the controllers is called the Input/Output (I/O) bus. The central portion of the I/O bus is a 16-bit wide, bidirectional shared data bus. Since this bus is shared by all the controllers as well as by the CPU, it is, by necessity, a half-duplex bus; i.e., only one operation can occur at any time. The direction of all information transfers on the I/O bus is defined relative to the computer. "Output" always refers to moving information from the computer to a controller; "input" always refers to moving information from a controller to the computer.

TYPES OF INFORMATION

The information transferred between a computer and a controller can be classified into three types: status, control, and data. Status information tells the computer about the state of the peripheral: is it busy?, is it ready?, is it operating properly? Control information is transferred by the computer to the controller to tell the peripheral what to do. Data is the information which originates from, or is sent to, the device during reading, writing, storing, or processing.

Irrespective of the type of information transferred, is the amount of information transferred. A single bit may be transferred; a collection of bits forming a byte (or character), 16 bits forming a word, or a group of words forming a block may be transferred.

TYPES OF INFORMATION TRANSFER

Information can be transferred between the computer and a peripheral in one of two ways: under direct program control or under data channel control. An information transfer occurring under direct program control moves a word or part of a word between an accumulator in the CPU and a register in the controller. This type of transfer occurs when an appropriate I/O instruction is executed in the program. An information transfer under data channel control generally moves a block of data, one word at a time, between the computer's memory and the device, through a register in the controller. The block of data is transferred automatically via the data channel once the program, using I/O instructions, sets up the transfer for a particular peripheral.

Direct Program Control

Direct program control of information transfers, also called "programmed I/O", is a way of transferring single words or parts of words to or from peripherals. Among the peripherals which transfer data in this way are terminals, paper tape readers and punches, card readers, line printers and plotters. Since the data moves through an accumulator, it is readily available to the program for manipulation or decision making. In the case of input, for example, the program can decide whether to read another word or character based on the value of the word or character just read.

However, because at least one instruction--and most likely several since the information must be stored in memory--must be executed for each character or word transferred, direct program control can be efficient only for peripherals which do not have to transfer large quantities of information quickly.

Data Channel Control

Some peripherals, such as discs and magnetic tape transports, are used to transfer large blocks of data. In order to reduce the amount of program overhead required, these blocks are transferred under data channel control. The commands used to set up the data channel transfer are assembled in an accumulator and are transferred to the controller under direct program control. The block of data is then automatically transferred directly between memory and the controller via the data channel.

Once the data channel transfer for a block of data has been set up and initiated by the program, no further action by the program is required to complete the transfer. The program can proceed with other tasks while the block transfer is taking place. Each time the controller is ready to transfer a word from the block it requests direct access to memory. When access is granted, the word is transferred. Because several instructions do not have to be executed for each word transferred, block transfers can occur at high rates, in some cases at more than a million words per second.

Since the actual transfer of a word via the data channel could conflict with the program instructions being executed, the program pauses during the transfer of each word. This pause is transparent to the programmer with the exception that the time required for program execution is lengthened.

PROGRAM INTERRUPT FACILITY

When transferring information under either direct program control or data channel control, the program must be able to determine when the transfer is complete, so that it can start a new transfer or proceed with a task that was dependent on the transfer just completed. Peripherals have status flags which can provide the program with this needed information. The I/O instruction set allows the program to check the status of these flags and perform decisions based on the results of the checks. However, these status checks are time-consuming, so, to avoid the necessity of continually performing such tests, all DGC computers incorporate a program interrupt facility.

The program interrupt facility provides a peripheral with a convenient means of notifying the processor that it requires service by the program. This is accomplished by allowing the peripherals to interrupt normal program flow on a priority basis. When a peripheral completes an operation or encounters a situation requiring processor intervention, it can request a program interrupt of the processor. The processor honors such a request by interrupting the program in process, saving the address where the interruption occurred, and transferring control to the interrupt handling routine. The interrupt handling routine can identify which peripheral requires service and transfer control to the service routine for that peripheral. After servicing that peripheral, the routine can restore the system to the state it was in when the interrupt occurred.

For computer systems which require large amounts of I/O to many devices, a multi-level priority structure up to 16 levels deep can be established. This structure can be set up to provide rapid service to those devices which are crucial to the efficient operation of the computer system; the less critical devices are serviced in as efficient a manner as possible. The priority interrupt structure, like the rest of the program interrupt facility, is under direct control of the program.

SUMMARY

The following sections of this Introduction to I/O Programming cover, in detail, the concepts introduced above. The instructions needed to perform a direct program controlled transfer are discussed in terms of their interaction with the controller and the CPU. The mechanics of the program interrupt facility together with methods used to arrange a priority structure are presented. Methods used in performing block transfers via the data channel are followed by a general discussion of the timing concepts which should be considered when designing an efficient system for handling I/O. Finally, examples are presented which illustrate the procedures discussed in this Introduction to I/O Programming.

This page intentionally left blank

I/O INSTRUCTION SET

INTRODUCTION

Information transfers between the computer and the various peripherals are governed by the program with eight instructions which constitute the I/O instruction set. These instructions allow the program to communicate with the peripherals' controllers and to control the program interrupt facility. This manual covers only those I/O instructions used for these purposes; additional I/O instructions used for special processor functions and options are fully described in the Programmer's Reference Manuals for the ECLIPSE™ and NOVA® line computers.

The effects of specific I/O instructions necessarily depend on the peripherals to which they are addressed. However, the general functions provided by the I/O instructions (loading and reading registers, issuing control signals, and testing flags) are the same for all peripherals; different peripherals merely use the available functions in different ways. In order to understand the general functions performed by the I/O instructions and how these functions are typically used by peripheral controllers, some knowledge of the architecture of a peripheral controller is required.

THE TYPICAL CONTROLLER

From the point of view of the program, a peripheral controller operates as a collection of information registers, control registers, and status flags, with which communications are established. With these registers and flags, the program can route data between the computer and the device and monitor the operation of the device. Information registers act as temporary depositories for information being transferred between the computer and the device. For an input operation, the device places information in a register in the controller and the computer then reads the register's contents. For an output operation, the computer places information in a register in the controller and the device can access that information when necessary. Control registers are loaded by the program and are used to control the operation of the device. Status flags are set by the peripheral to reflect its current operating conditions. The program, through the use of I/O instructions can examine these status flags and can alter some of them to change the operating state of the peripheral.

The distinction made here between registers and flags is generally one of information content. A flag contains a single bit of information, while a register is made up of a number of bits. Groups of contiguous bits in a register which convey a single "piece" of information are referred to as "fields". For example, in one of the magnetic tape controller's registers, bits 13-15 act together as a control field to select one of the eight possible tape transports in the subsystem.

The paragraphs below describe only the basic components of a typical controller. The additional structure required for a peripheral using the program interrupt facility or the data channel is discussed in the sections describing those facilities. What follows is meant only to typify the workings of a controller; controllers are tailored to the specific devices they control, so that not all fit the model given here.

Information Registers

A controller usually contains one or more information registers. Using I/O instructions, the program can load data and control information into these registers from the processor's accumulators and can read data and status information into the accumulators from them. The three types of information considered here--data, control, and status--give rise to three basic types of information registers, which are described below.

Data Registers

A data register (or data buffer) is used to store data in the controller as it passes between the device and the computer. This buffer is needed because the computer and the device usually operate at different speeds. Since the operation of nearly all peripherals involve the transfer of a word or part of a word of data between the computer and the device, nearly all peripherals controllers contain a data buffer. In the case of peripherals which transfer data under direct program control, the data buffer is directly accessible to the program. Data is transferred between the register in the controller and an accumulator in the central processor by an I/O instruction. In the case of a peripheral which transfers data under data channel control, the data is transferred directly between the register in the controller and memory. Data buffers in the controllers which use the data channel need not be--and usually are not--accessible to the program.

Control Registers

Control registers allow the program to supply the controller with information necessary for the operation of the device, such as drive or transport numbers, data block sizes, and command specification. A unit of control information is called a "control parameter". Control parameters typically allow the program to select one of a number of peripheral units in a subsystem, the operation to be performed, and the initial values for flags and counters in the controller. The program specifies control parameters to the controller with an I/O instruction wherein the desired parameters are coded into the appropriate fields of the accumulator used in the transfer.

Status Registers

Status registers are used to indicate to the program the detailed state of the peripheral. They consist primarily of status flags, but can also contain control parameters. The control parameters contained in status registers are commonly those which change during the operation of the peripheral, and are therefore of importance to the program which must check on the progress of the peripheral's operation. For example, a program transferring consecutive sectors of information on a disc in a single operation can read the current sector address and sector count during the operation in order to determine how far the operation is from completion. Status flags are set by the controller to indicate error conditions or to notify the computer of the basic state of the peripheral.

The classification of controller registers into the three types described above can only be a general one. A register may contain more than one type of information. The most common case of this occurrence is a register that serves as a control register when loaded by the program and as a status register when read by the program. The disc address/sector counter register mentioned in the preceding paragraph is an example of such a combined control and status register.

Busy and Done Flags

The Busy and Done flags are the two fundamental flags in a controller and they serve a dual purpose. Together they denote the basic state of the peripheral and can be tested by the program to determine that state. In addition, the program can manipulate these flags in order to control the operation of the peripheral. To place the peripheral in operation, the program sets the Busy flag to 1. The Busy flag remains in this state for the duration of the operation, indicating that the peripheral is in use and should not be disturbed by the program. When the peripheral completes its operation, the controller sets the Busy flag to 0 and the Done flag to 1 to indicate this fact. The setting of the Done flag to 1 can be used to trigger a program interrupt. Whether a program interrupt occurs depends on the state of the interrupt facility. However, no matter what state the interrupt facility is in, no interrupt can occur for that peripheral until its Done flag is set to 1. Therefore, the setting to 1 of the Done flag is defined to "initiate a program interrupt request". At this point, the program can either start the next operation by setting the Done flag to 0 and the Busy flag to 1, or it can idle (clear) the peripheral by setting both flags to 0.

Other Status Flags

For a relatively simple peripheral, the Busy and Done flags alone may furnish enough status information to allow the program to service the peripheral adequately. However, a more complex peripheral will generally require additional status flags to specify its internal operating conditions more completely to the program. The difference between these additional status flags and the Busy and Done flags lies in the way the program tests them. The program can test the Busy and Done flags directly with a single I/O instruction, but checking any other status flag requires that its value first be read into an accumulator from the status register. Each status flag is assigned by the controller to one of the 16 available bit positions in the status register. The program may then perform any test it requires on the status word after it is read.

Error Flags

Status flags which indicate errors or malfunctions in the operation of a peripheral are termed "error flags". Two types of error flags can be characterized, according to their effect on the operation of the peripheral when they are set. The first, or passive, type is merely set by the controller in the course of the operation when the associated error occurs. No immediate indication of this type of error is given to the program, and the operation is allowed to continue to completion. The second, or active, type of error flag is set by the controller when the program attempts to start an operation which is not allowed. In this case, the operation never begins and the Done flag is set to 1 immediately to notify the program. This type of error flag is used to prevent a severe and probably irrecoverable error from occurring. In either case, the program must respond, error or not, when it notices that a peripheral is "done". It need only check the appropriate error flag or flags before assuming that the operation it initiated was satisfactorily completed.

For example, among its many status flags, the controller for magnetic tape transports contains error flags to indicate parity errors and illegal operations. During a read operation, when a character is read with incorrect parity, the Parity Error flag is set to 1. No immediate notification of the error is given to the program and the read operation is allowed to finish. The parity error can be detected at the completion of the operation, when the program should check for errors. At this time appropriate action can be taken, such as trying to read the misread section of tape again or printing an error message on the console terminal. The Illegal flag, on the other hand, which is set when an illegal operation is attempted, prevents the operation from starting. The controller immediately sets both the Done and Illegal flags to 1 to notify the program. Illegal operations for a magnetic tape transport include writing on a tape that is write-protected and spacing backwards when the tape is at the beginning of tape marker.

INSTRUCTION FORMAT

The general format of the I/O instructions is shown below.

0	1	1	AC		OP CODE			CONTROL		DEVICE CODE					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-2 are 011 and identify this as an I/O instruction, bits 3-4 specify an accumulator, bits 5-7 contain the operation code, bits 8-9 specify a flag control function or test condition, and bits 10-15 specify the code of the device.

Device Code Field

Bits 10-15 in an I/O instruction select the peripheral that is to respond to the instruction. The instruction format thus allows for 64 device codes, numbered 0-77_8. In all computers, device code 0 is not assigned to any peripheral, and device code 77_8 is used to implement a number of specific processor functions, such as reading the console switches and controlling the program interrupt facility. Depending on the computer, a number of other specific device codes are reserved for processor options or features. The remaining device codes are available for referencing peripherals. Many of these codes have been assigned by Data General Corporation to standard peripherals, and the assembler recognizes convenient mnemonics for these codes. The list of the standard device code assignments and their associated mnemonics is given in Appendix A.

Flag Control Field

The Busy and Done flags are either manipulated or tested by the control functions or test conditions specified in bits 8 and 9 of the I/O instructions. In those instructions which allow flag manipulation, bits 8 and 9 are referred to as the F field. The flag control commands available, along with the associated mnemonics and bit configurations and the functions typically performed, are as follows:

F field	Command	Mnemonic	Control Function
00	(none)	(omitted)	None
01	Start	S	Start the peripheral by setting the Busy flag to 1 and the Done flag to 0.
10	Clear	C	Clear (idle) the peripheral by setting both the Busy and Done flags to 0.
11	Pulse	P	Pulse the controller to achieve a special effect. The effect, if any, depends on the peripheral.

In the I/O instruction which allows flag testing, bits 8 and 9 are referred to as the T field. The bit configurations, mnemonics, and test conditions they select are as follows:

T field	Mnemonic	Next instruction is skipped if:
00	BN	Busy flag is 1 (Non-zero)
01	BZ	Busy flag is 0 (Zero)
10	DN	Done flag is 1 (Non-zero)
11	DZ	Done flag is 0 (Zero)

Two important features of the I/O instruction set result from the nature of the flag control field. First, because the flag control field is separate from the operation code field, a single I/O instruction can both transfer information between the controller and the computer and simultaneously control the operation of the peripheral. Secondly, the use of the flag control field as a T field allows the direct testing of a controller's Busy or Done flag in a single instruction, so that quick decisions based on the basic state of the peripheral can be made by the program.

Operation Code Field

The 3-bit operation code field selects one of the eight I/O instructions. In two of these instructions, no information transfer is specified; instead, bits 8 and 9 may specify either a control function or a flag test condition as described above. The remaining six instructions involve an information transfer between the computer and the designated peripheral controller and may also specify a control function to be performed after the information transfer has been completed. The program can, therefore, access up to six registers in any one controller. Up to three of these six registers are output registers which can be loaded by the program with either data or control information. The other three are input registers, from which the program can read either data or status information. It is entirely possible and, in fact, quite common for two different I/O instructions, one input and one output, to reference the same register in a controller. However, this is not in any way required by the nature of the I/O instruction set; potentially six different registers in a controller may be accessible to the program.

In order to give names and mnemonics to the I/O instructions in their general form, the registers in a peripheral controller which are accessible to the program are referred to with letter designations. The three input registers are called the "A input buffer", the "B input buffer", and the "C input buffer". Similarly, the three output registers are called the "A output buffer", the "B output buffer", and the "C output buffer". Thus, for example, to read data from a peripheral controller's A input buffer, a DATA IN A instruction, with mnemonic DIA, is issued to that peripheral.

The eight operation code fields, their associated mnemonics, and the instructions specified are as follows:

Operation Code field	Mnemonic	Instruction
000	NIO	No Input or Output but perform the flag control function specified.
001	DIA	Read Data Into the computer from the A input buffer.
010	DOA	Write Data Out from the computer to the A output buffer.
011	DIB	Read Data Into the computer from the B input buffer.
100	DOB	Write Data Out from the computer to the B output buffer.
101	DIC	Read Data Into the computer from the C input buffer.
110	DOC	Write Data Out from the computer to the C output buffer.
111	SKP	SKiP the next instruction if the test selected for the Busy or Done flag is true.

Accumulator Field

Bits 3 and 4 in an I/O instruction select one of the central processor's four accumulators: AC0, AC1, AC2, or AC3. In those instructions which involve an information transfer between the processor and a peripheral controller, the specified accumulator either furnishes the information for an output transfer or receives the information in an input transfer. In the two I/O instructions which do not involve an information transfer, the accumulator field is ignored. The assembler sets bits 3 and 4 in these instructions to 0; however, any bit combination will do, and no accumulator will ever be affected by these two instructions.

INSTRUCTIONS

A number of abbreviations and symbols are used in this manual to aid in defining how an instruction may be coded in assembly language. Abbreviations used are as follows:

AC or ac	accumulator
F or f	flag control command
T or t	flag test command
device	device code or mnemonic

The following symbols are not coded, rather they perform these functions:

< > Indicates an optional operand. The operand enclosed in the brackets (e.g., <f>) may be coded or not, depending on whether the associated option is desired.

= Indicates a specific substitution is required. Substitute the desired number, letter or letters, or symbols from the class, as defined by the abbreviation for which the substitution is being made. For example, "ac" indicates that an accumulator specifier is required. To select AC2, code either a "2" or a symbol whose value is 2.

When describing the format of a word involved in an information transfer between the computer and a controller, the various fields and bits in the word are labeled with names descriptive of their functions. Bits in the word which are not used by the controller are shaded. Shaded bits are ignored on output and set to 0 on input.

NO I/O TRANSFER

NIO<f> device

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	0	0	F		DEVICE CODE					

The Busy and Done flags in the controller of the specified device are set according to the function specified by F. When the assembler encounters the mnemonic NIO, it sets the AC field bits to 0. However, these bits are ignored and may have any value. The contents of all the accumulators are unchanged.

I/O SKIP

SKP<t> device

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	1	1	T		DEVICE CODE					

Skip the next sequential instruction if the test condition specified by T is true for the specified controller. When the assembler encounters the mnemonic SKP<t>, it sets the AC field bits to 0. However, these bits are ignored and may have any value. The contents of all the accumulators and the Busy and Done flags for the specified device remain unchanged.

DATA IN A

DIA<f> ac,device

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	0	1	F		DEVICE CODE					

The contents of the A input buffer in the specified controller are placed in the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The number of information bits transferred depends on the size of the buffer and the mode of operation of the peripheral. Bits in the specified AC that do not receive information are set to 0.

DATA OUT A

DOA<f> ac,device

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	0	F		DEVICE CODE					

The contents of the specified AC are placed into the A output buffer in the specified controller. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The number of information bits loaded into the buffer depends on the size of the buffer and the mode of operation of the peripheral. Any unused bits are ignored by the controller. The contents of the specified AC remain unchanged.

DATA IN B

DIB<f> ac,device

0	1	1	AC		0	1	1	F		DEVICE CODE					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the B input buffer in the specified controller are placed in the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The number of information bits transferred depends on the size of the buffer and the mode of operation of the peripheral. Bits in the AC that do not receive information are set to 0.

DATA OUT B

DOB<f> ac,device

0	1	1	AC		1	0	0	F		DEVICE CODE					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the specified AC are placed in the B output buffer in the specified controller. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The number of information bits loaded into the buffer depends on the size of the buffer and the mode of operation of the peripheral. Any unused bits are ignored by the controller. The contents of the specified AC remain unchanged.

DATA IN C

DIC<f> ac,device

0	1	1	AC		1	0	1	F		DEVICE CODE					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the C input buffer in the specified controller are placed in the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The number of information bits transferred depends on the size of the buffer and the mode of operation of the peripheral. Bits in the AC that do not receive information are set to 0.

DATA OUT C

DOC<f> ac,device

0	1	1	AC		1	1	0	F		DEVICE CODE					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the specified AC are placed in the C output buffer of the specified controller. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The number of information bits loaded into the buffer depends on the size of the buffer and the mode of operation of the peripheral. Any unused bits are ignored by the controller. The contents of the specified AC remain unchanged.

The single letter mnemonic for the desired control command is appended to the basic mnemonic. An NIO instruction alone with any device code is a "no-op" (that is, it has no effect). To place the high speed paper tape reader in operation, an NIOS 12, or, using the reader's mnemonic, an NIOS PTR instruction could be executed. Both of these instructions assemble as 060112_8 (0 110 000 001 001 010) and cause the reader to read one character from the tape into the controller's data buffer.

To determine when the character is in the buffer without using the program interrupt we can wait for either the Busy flag to be set to 0 or the Done flag to be set to 1. For example, giving

```
SKPDN   PTR
JMP     .-1
```

keeps the processor from proceeding until the reader controller has set the Done flag to 1.

Once the character has been loaded into the data buffer, the Done flag is set to 1 and the processor will continue. The character can be read into the right half of AC2 by giving

```
DIA   2,PTR
```

If another character is to be read the transfer can be made with a

```
DIAS   2,PTR
```

which brings the character into AC2, sets the Done flag to 0, the Busy flag to 1, and causes the reader to read the next character. If the data buffer contains the final character to be read from tape, giving

```
DIAC   2,PTR
```

retrieves the character and sets both the Busy and Done flags to 0, thus idling the reader.

In the remainder of this manual the discussion of each peripheral treats only the control functions and the applicable I/O transfer instructions. The skips apply to all and are the same in all cases. Giving a data-in instruction that does not apply to a peripheral (either because the peripheral is output-only or does not have the buffer specified) sets all bits of the addressed accumulator to 0 but the specified flag control function is carried out. Similarly, a data-out instruction that does not apply is a no-op except for the specified control functions. When the device code is undefined or the addressed peripheral is not in the system, any data-out or NIO instruction, a SKPBN, or a SKPDN is a no-op; a SKPBZ or SKPDZ is an unconditional skip (equivalent to a JMP .+2); and any data-in instruction simply sets all bits of the specified AC to 0.

This page intentionally left blank

PROGRAM INTERRUPT FACILITY

INTRODUCTION

When a peripheral completes an operation, the controller sets its Done flag to 1 to indicate that program service is required. The program can test the state of the Done flag repeatedly with I/O SKIP instructions to determine when this occurs. However, continual interrogation of the Done flag by the program is generally wasteful of computing time, especially when flag checks need to be done frequently in order to ensure that service is not delayed so long that the peripheral loses data. The program interrupt facility provides a peripheral with a convenient means of notifying the processor that service is required.

All peripherals which use the program interrupt facility have access to a single direct line to the processor, called the Interrupt Request Line, along which their requests for service are communicated. An interrupt request can be generated by a peripheral when the peripheral's Done flag is set to 1. The processor can respond to, or "honor", an interrupt request by interrupting the normal flow of program execution and transferring control to an interrupt handling routine. The programmer can control which peripherals may request interrupts and when the processor may start an interrupt, by manipulating a number of flags which are distributed among the processor and the peripherals.

The operation of the program interrupt facility, as controlled by these flags, is described below. Following portions of this section detail the instructions used to control the program interrupt facility, describe the implementation of a priority interrupt scheme, offer further suggestions for programming an interrupt handler, and explain the operation of the vector instruction, which allows the ECLIPSE computer to automatically perform much of its interrupt processing.

OPERATION

Control Flags

The operation of the program interrupt facility is governed by the Interrupt On (ION) flag in the central processor and by the Done and Interrupt Disable flags in each peripheral which uses the facility. By manipulating these flags, the program can choose to disregard interrupt requests altogether, or it can selectively ignore certain peripherals.

Interrupt On Flag

The major control flag for the program interrupt facility is the Interrupt On flag in the central processor. To enable the interrupt facility the program sets ION to 1, allowing the processor to respond to interrupt requests transmitted to it along the Interrupt Request Line. Setting ION to 0 disables the entire interrupt facility, causing the processor to ignore all interrupt requests.

The Interrupt On flag is manipulated by the program exactly like a Busy flag for the central processor. A Start command in any I/O instruction directed to the CPU (device code 77_8) sets ION to 1, a Clear command in such an instruction sets ION to 0. (ION is also set to 0 at power-up and when the RESET console switch is pressed.)

Interrupt Disable Flags

The controller for each peripheral which uses the program interrupt facility contains an Interrupt Disable flag which allows the program to disable interrupts from that peripheral. When a peripheral's Interrupt Disable flag is set to 1, the peripheral is prevented from making an interrupt request.

The Interrupt Disable flags of all peripherals are manipulated at once with a single I/O instruction. This instruction, MASK OUT (MSKO), sets up the Interrupt Disable flags of all peripherals connected to the program interrupt facility according to a mask contained in the accumulator specified by the instruction. Each peripheral is assigned by its hardware to a bit position in the mask. (Mask bit assignments for standard peripherals are given in Appendix A.) When a MASK OUT instruction is given, each peripheral's Interrupt Disable flag is set to the value of the assigned bit of the mask. Also, at power-up and when the RESET console switch is pressed, all Interrupt Disable flags are set to 0.

Interrupt Requests

Interrupt requests by a peripheral are governed by its Done and Interrupt Disable flags. When a peripheral completes an operation, it sets its Done flag to 1, and this action initiates a program interrupt request. If its Interrupt Disable flag is 0, the request is communicated to the CPU. If the Interrupt Disable flag is 1, the request is not communicated to the CPU; it is blocked until the Interrupt Disable flag is set back to 0. If the ION flag is 1, the processor has to honor the interrupt request as soon as it is able.

The processor is able to interrupt the sequential flow of program instructions if all of the following conditions hold:

1. The processor has just completed an instruction or a data channel transfer occurring between two instructions.
2. At least one peripheral is requesting an interrupt.
3. Interrupts are enabled; that is, ION is 1.
4. No peripheral is waiting for a data channel transfer; that is, there are no outstanding data channel requests. The data channel has priority over program interrupts.

When the processor finishes an instruction it takes care of all data channel requests before it starts an interrupt; this includes any additional data channel requests that are initiated while data channel transfers are being made. When no more peripherals are waiting for data channel transfers, the processor starts an interrupt if ION is 1 and at least one peripheral is requesting an interrupt.

Processor Response

The processor starts an interrupt by automatically executing the following sequence:

1. It sets ION to 0 so that no further interrupts may be started.
2. It stores the contents of the program counter (which point to the next instruction in the interrupted program) in location 0, so that a return to the interrupted program can be made after the interrupt service routine has finished.
3. It simulates a JMP@1 instruction to transfer control to the interrupt service routine. Location 1 should contain the address of the routine or the first part of an indirect address chain that points to the routine.

Servicing An Interrupt

The interrupt service routine (or handler) should save the state of the processor, identify which peripheral requires service, and service the peripheral.

Saving the state of the processor involves saving the contents of any accumulators that will be used in the interrupt service routine and saving the carry bit if it will be used. The state of the processor must be saved so that it may be restored before the interrupted program is allowed to resume.

Peripheral Identification

There are three ways in which the interrupt handler can identify which peripheral requires service.

1. On the NOVA line and ECLIPSE computers, the interrupt handler can execute a polling routine. This routine is merely a sequence of I/O SKIP instructions which test the states of the Done flags of all peripherals in use. With this method peripheral priorities are determined by the order in which the tests are performed. Note that the polling technique disregards the state of the Interrupt Disable flags. Peripherals which are masked out will be recognized if their Done flags are 1, even though these peripherals could not have caused the interrupt.
2. On the NOVA line and ECLIPSE computers, the interrupt handler can issue an INTERRUPT ACKNOWLEDGE instruction (INTA). This instruction reads the device code of the first peripheral on the I/O bus that is requesting an interrupt, into a specified accumulator. Note that with this method the Interrupt Disable flags are significant. Peripherals which are masked out cannot request an interrupt and, therefore, cannot respond to the INTERRUPT ACKNOWLEDGE instruction.
3. On the ECLIPSE computer, the interrupt handler can issue a VECTOR instruction (VCT). This instruction determines which peripheral requires service in exactly the same way as the INTERRUPT ACKNOWLEDGE instruction. However, the device code obtained is not placed in an accumulator but is used as an index into a table of addresses. Besides vectoring automatically to the correct peripheral service routine, the VECTOR instruction can optionally switch stack contexts, save the state of the processor, and establish a new priority level. Because the VECTOR instruction is available only on the ECLIPSE computer, and because its operation is relatively complex, it is describea later in a section of its own.

Peripheral Service Routine

After determining which peripheral requires service, the interrupt handler generally transfers control to a peripheral service routine. This routine performs the information transfer to or from that peripheral (if required) and either starts the peripheral on a new operation or idles the peripheral if no more operations are to be performed at this time.

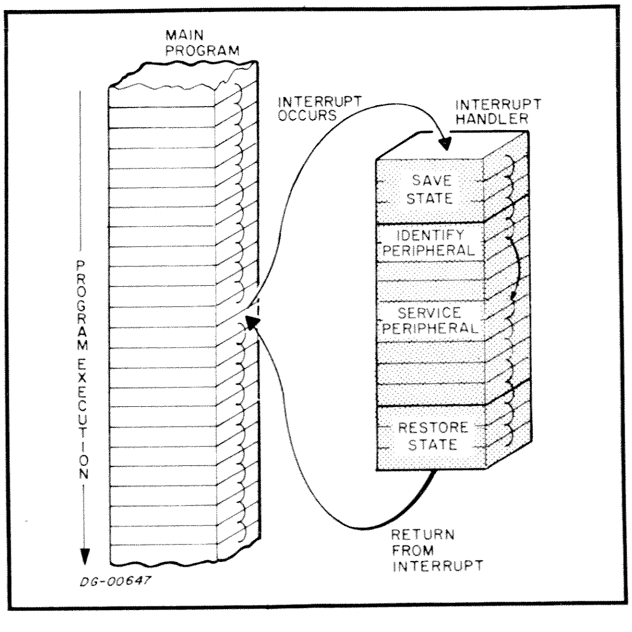
Dismissing An Interrupt

When all service for the peripheral has been completed, either the peripheral service routine or the main interrupt handler should perform the following sequence to dismiss the interrupt.

1. Set the peripheral's Done flag to 0 to dismiss the interrupt request which was just honored. If this is not done, the undismissed interrupt request will cause another interrupt--this time incorrectly--as soon as the interrupt handler finishes and attempts to return control to the interrupted program.
2. Restore the pre-interrupt states of the accumulators and the carry bit.
3. Set the Interrupt On flag to 1 to enable interrupts again.
4. Jump back to the interrupted program. (Usually a JMP@0 instruction is given.)

The instruction that enables interrupts (usually INTEN) sets the Interrupt On flag to 1, but the processor does not allow the state of the ION flag to change to 1 until the next instruction begins. Thus, after the instruction that turns interrupts back on, the processor always executes one more instruction (assumed to be the return to the interrupted program) before another interrupt can start. The program must give this final return instruction immediately after enabling interrupts in order to ensure that no waiting interrupt can overwrite the contents of location 0 before they are used to return control to the interrupted program.

The following diagram shows how normal program flow is altered during a program interrupt. The interrupt handler is shaded to indicate that this block of instructions is not interruptable since the processor sets the ION flag to 0 to disable further interrupts when the interrupt occurred. Interrupts are not enabled again until the interrupt handler executes its INTERRUPT ENABLE instruction just prior to returning control to the interrupted program.



INSTRUCTIONS

The instructions which control the program interrupt facility use special device code 77_8 (mnemonic CPU). When this device code is used, bits 8 and 9 of the skip instructions test the state of the Interrupt On flag; in the other instructions these bits turn interrupts on or off by setting ION to 1 (Start command) or 0 (Clear command).

INTERRUPT ENABLE

INTEN

NIOS CPU

0	1	1	AC		0	0	0	0	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Set the Interrupt On flag to 1 to allow the processor to respond to interrupt requests. If the Interrupt On flag actually changes state (from 0 to 1), the processor will execute one more instruction before it can start an interrupt. On the ECLIPSE computer, the processor will execute one more instruction before starting an interrupt even if the Interrupt On flag was already 1. However, if that instruction is one of those that is interruptable, then an interrupt can occur as soon as the instruction begins to execute. The assembler recognizes the mnemonic INTEN as equivalent to NIOS CPU.

INTERRUPT DISABLE

INTDS

NIOC CPU

0	1	1	AC		0	0	0	1	0	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Set the Interrupt On flag to 0 to prevent the processor from responding to interrupt requests. The assembler recognizes the mnemonic INTDS as equivalent to NIOC CPU.

SKIP IF INTERRUPTS ENABLED

SKPBN CPU

0	1	1	AC		1	1	1	0	0	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Skip the next sequential instruction if the Interrupt On flag is 1.

SKIP IF INTERRUPTS DISABLED

SKPBZ CPU

0	1	1	AC		1	1	1	0	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Skip the next sequential instruction if the Interrupt On flag is 0.

MASK OUT

MSKO ac

DOB<f> ac,CPU

0	1	1	AC		1	0	0	F		1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Set the Interrupt Disable flags in all peripherals according to the mask contained in the specified AC. (A 1 in a mask bit sets the flags in all peripherals assigned to that bit to 1, a 0 sets them to 0.) After the Interrupt Disable flags are set, the Interrupt On flag is set according to the function specified by F. The contents of the specified AC remain unchanged. Mask bit assignments for standard peripherals are given in Appendix A. The assembler recognizes the instruction MSKO ac as equivalent to DOB ac,CPU.

INTERRUPT ACKNOWLEDGE

INTA ac

DIB<f> ac,CPU

0	1	1	AC		0	1	1	F		1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The device code of that peripheral requesting an interrupt which is closest to the processor along the I/O bus is placed in bits 10-15 of the specified AC. Bits 0-9 are set to 0. After the data transfer, the Interrupt On flag is set according to the function specified by F. If no peripheral is requesting an interrupt, the specified AC is set to 0. The assembler recognizes the instruction INTA ac as equivalent to DIB ac,CPU.

I/O RESET

IORST

DIC<f> ac,CPU

0	1	1	AC		1	0	1	F		1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Reset all peripherals connected to the I/O bus: set their Busy, Done, and Interrupt Disable flags to 0 and, depending on the peripheral, perform any other required initialization. After the peripherals' flags are altered, the Interrupt On flag is set according to the function specified by F. The assembler recognizes the mnemonic IORST as equivalent to DICC 0,CPU--that is, as the instruction defined here with F set to 10.

If the mnemonic DIC is used to perform this function, an accumulator must be coded to avoid assembly errors. Regardless of how the instruction is coded, during execution the AC field is ignored and the contents of the specified AC remain unchanged. At power-up and when the RESET console switch is pressed, the processor performs the equivalent of an IORST instruction.

The assembler recognizes a number of convenient mnemonics for instructions that control the program interrupt.

Mnemonic	Instruction	Mnemonic Equivalent	Octal Equivalent
INTEN	INTERRUPT ENABLE	NIOS CPU	060177
INTDS	INTERRUPT DISABLE	NIOC CPU	060277
MSKO ac	MASK OUT	DOB ac,CPU	062077
INTA ac	INTERRUPT ACKNOWLEDGE	DIB ac,CPU	061477
IORST	I/O RESET	DICC 0,CPU	062677

To set up the Interrupt Disable flags according to the mask contained in AC2, give

MSKO 2
or
DOB 2,CPU

However, there is one important difference between these special mnemonics and the standard ones: mnemonics for enabling and disabling interrupts cannot be appended to them. Thus, to set the Interrupt On flag to 0 while performing a MASK OUT instruction using AC2 give

DOBC 2,CPU

Note that use of the mnemonic IORST sets the Interrupt On flag to 0. To set the flag to 1 while resetting the peripherals give

DICS 0,CPU

PRIORITY INTERRUPTS

If the Interrupt On flag remains 0 throughout the interrupt service routine, the routine cannot be interrupted, and there is only one level of peripheral priority. All peripherals that have not been disabled by the program are, for the most part, equally able to request interrupts and receive interrupt service. Only when two or more peripherals are requesting an interrupt at exactly the same time is a priority distinction made. When this happens, priority is determined either by the order in which I/O SKIP instructions are given or, if the INTERRUPT ACKNOWLEDGE or VECTOR instruction is used, by the order of peripherals along the I/O bus. In a system with peripherals of widely differing speeds and/or service requirements, a more extensive priority structure may be required. The program interrupt facility hardware and instructions allow the program to implement up to 16 interrupt priority levels.

For example, suppose that a card reader and a Teletype® are being operated at the same time. While a card is being read, an interrupt is requested as each new column of data is available, and the program must read this data within 430 microseconds, typically, before it is overwritten in the Data Buffer by the data from the next column. If the Teletype service routine takes 300 microseconds, card reader service will never be delayed longer than this, and a single-level program interrupt scheme will suffice. However, this interrupt scheme will not work if the Teletype service routine takes 600 microseconds, since a card reader interrupt request initiated soon after Teletype service is begun will not be honored in time, and a column of data will be lost. In order to avoid losing data, the program interrupt scheme used must allow the card reader to interrupt the lengthy Teletype service routine. This involves creating a two-level priority structure and assigning the card reader to the higher priority level.

In general, a multiple-level priority interrupt scheme is used to allow higher-priority peripherals to interrupt the service routines of lower-priority peripherals. A hierarchy of priority levels can be established through program manipulation of the Interrupt Disable flags of all peripherals in the system. When the interrupt request from a peripheral of a certain priority is honored, the interrupt handler sets up the new priority level by establishing new values for all peripherals' Interrupt Disable flags according to an appropriate "Interrupt Priority Mask" used with the MASK OUT instruction. Peripherals whose Interrupt Disable flags are set to 1 by the corresponding bit of this priority mask are "masked out", or disabled, and are thereby regarded as being of lower priority than the peripheral being serviced. Peripherals which are not masked out assume a higher priority than the

Teletype® is a registered trademark of Teletype Corporation, Skokie, Illinois. All references to teletypes in this manual shall apply to this mark.

peripheral being serviced. Before proceeding with the peripheral service routine, the Interrupt On flag is set to 1 so that the higher-priority peripherals may interrupt the current service routine.

Interrupt Priority Mask

The bit of the priority mask that governs the Interrupt Disable flag for a given peripheral is assigned to that peripheral by the hardware and cannot be changed by the program. Although lower-speed devices are generally assigned to higher-numbered mask bits, no implicit priority ordering is intended. The manner in which these priority levels are ordered is completely up to the programmer. By means of the priority mask the program can establish any desired priority structure, with one limitation: in the cases in which two or more peripherals are assigned to the same bit of the priority mask, these peripherals are constrained to be at the same priority level. When a peripheral causes an interrupt, a decision must be made whether to place all other peripherals which share the same mask bit with the interrupting peripheral at a higher or lower priority level. If a decision is made to mask out all peripherals which share that priority mask bit, the interrupting peripheral is also masked out.

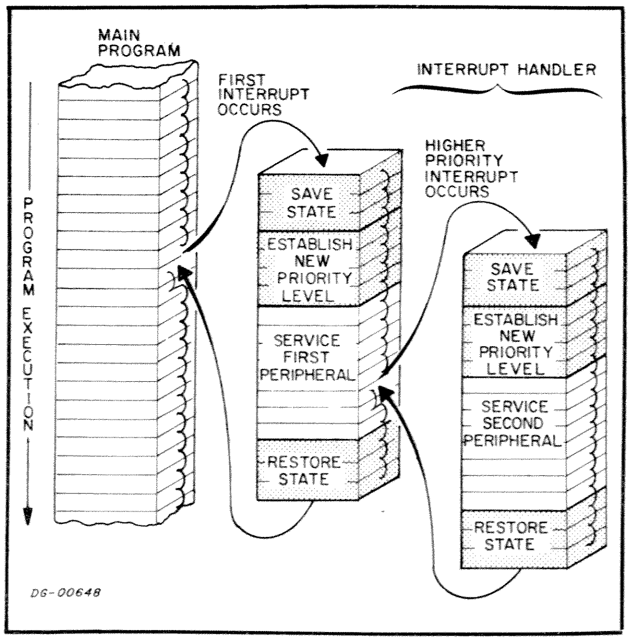
Priority Interrupt Handler

A priority interrupt handler differs from a single-level interrupt handler in several ways. The handler must be "re-entrant". This means that if a peripheral service routine is interrupted by another, higher priority peripheral, no information required by the handler to restore the state of the machine, is lost. The two items of information which should be saved, in addition to those saved by a single-level interrupt handler, are the return address (the contents of location 0) and the current priority mask. This information must be stored in different locations each time the interrupt handler is entered at a higher level. Doing this ensures that the necessary return information belonging to an earlier interrupt is not overwritten by a higher level interrupt. A common method of storing return information for a re-entrant interrupt handler is through the use of push-down stacks.

The interrupt handler (including the peripheral service routines) for a multi-level priority scheme should perform the following tasks:

1. Save the state of the processor, that is, the contents of the accumulators, the carry bit, location 0, and the current priority mask.
2. Identify the peripheral that requested the interrupt.
3. Transfer control to the service routine for that peripheral.
4. Establish the new priority mask with a MASK OUT instruction for that peripheral's service routine and store it in memory at the location reserved for the current priority mask for that level of interrupt.
5. Enable interrupts. Now, any peripheral not masked out can interrupt this service routine.
6. Service the peripheral that requested the interrupt.
7. Disable interrupts in preparation for dismissal of this interrupt level, so that no interrupts will occur during the transition to the next lower level.
8. Restore the state of the processor, including the former contents of the accumulators and the carry bit and reinstitute the pre-interrupt priority mask with a MASK OUT instruction.
9. Enable interrupts.
10. Transfer control to the return address which was saved from location 0.

The diagram below is a simplified representation of program flow in a priority interrupt environment. Shaded areas indicate non-interruptable sections of instructions. Additional higher-priority interrupts could increase the depth of interrupts still further.



The Vector Instruction

The ECLIPSE computer incorporates an instruction which greatly reduces the burden of programming a priority interrupt system. Since this instruction is available only on the ECLIPSE computer, it is described separately below. In effect, the VECTOR instruction (VCT) can be used to perform the first five tasks listed above for the multi-level priority interrupt handler.

VECTOR ON INTERRUPTING DEVICE CODE

VCT <@>displacement

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
@	DISPLACEMENT														

This instruction provides a fast and efficient method for transferring control from the main I/O interrupt handler to the correct interrupt service routine for the interrupting device. Bit 0 of the second word of the instruction is the "stack change bit" and bits 1-15 contain the address of a 64-word vector table. Vector table entries are one word in length and consist of a "direct" bit in bit 0 followed by an address in bits 1-15.

An INTERRUPT ACKNOWLEDGE instruction is performed. The device code returned is added to the address of the vector table and the vector table entry at that address is fetched. If the direct bit in the fetched vector table entry is 0, the address in bits 1-15 is taken to be the address of the device handler routine for the interrupting device and control is immediately transferred there by placing the address in the program counter.

If the direct bit is 1, the address in bits 1-15 of the vector table entry is taken to be the address of the device control table (DCT) for the interrupting device. At this point, the stack change bit is examined. If the stack change bit is 0, no stack change is performed. If the stack change bit is 1, a new stack is created by placing the contents of memory location 6 in the stack limit, and the contents of memory location 7 in the stack fault. The previous contents of memory locations 40_8-43_8 are then pushed onto this new stack.

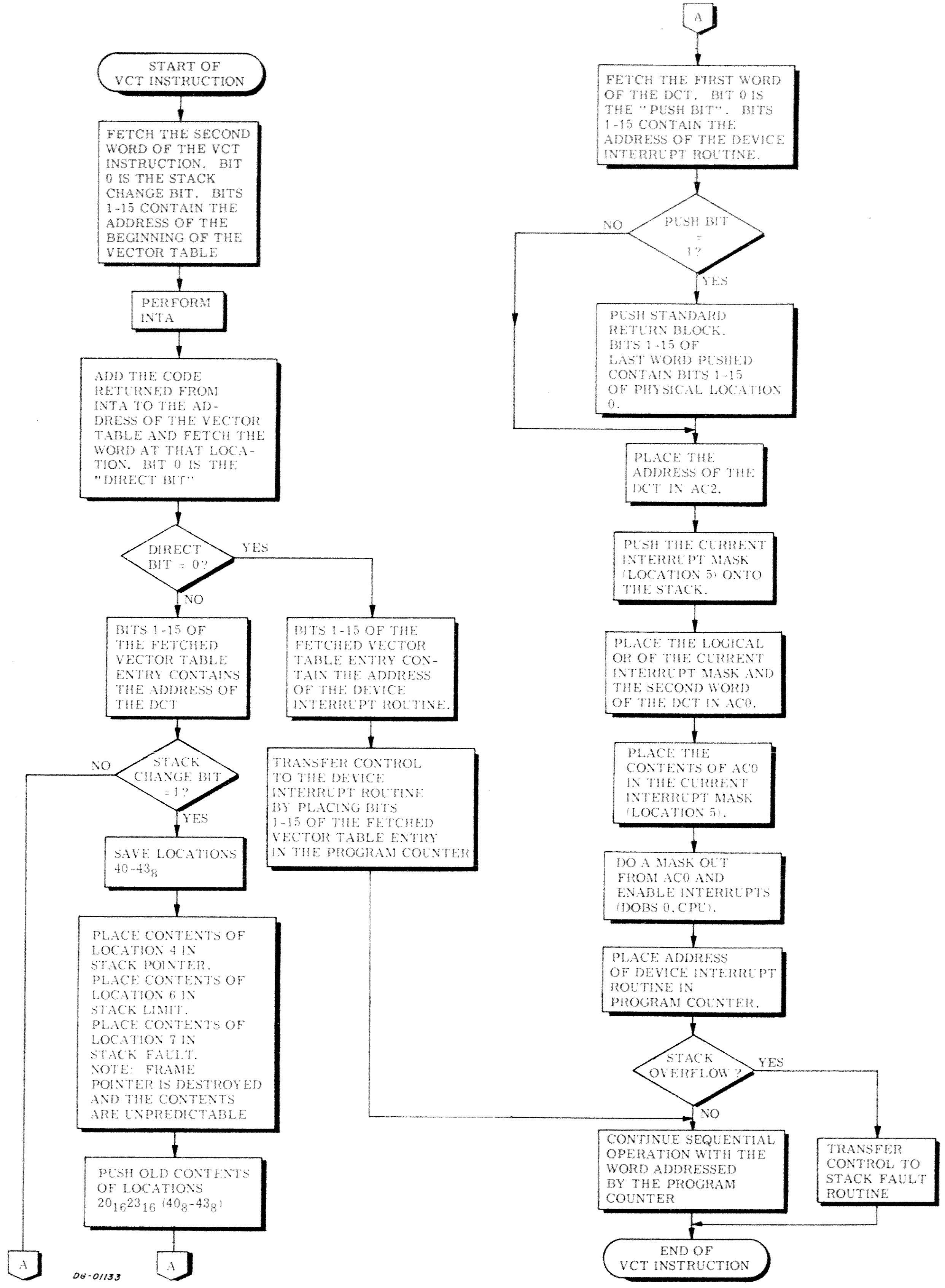
Device control tables must consist of at least two words. The first word of a DCT consists of a "push bit" in bit 0 followed by the address of the device handler routine for the interrupting device in bits 1-15. The second word of a DCT contains a mask that will be used to construct the new interrupt priority mask. Succeeding words in a DCT may contain information that is to be used by the device interrupt handler.

After the stack change procedure is performed, the first word of the DCT is fetched and inspected. If the push bit is 1, a standard return block is pushed onto the stack with bits 1-15 of physical location 0 placed in bits 1-15 of the last word pushed. If the push bit is 0, no return block is pushed.

Following this procedure, the address of the DCT is placed in bits 1-15 of AC2 and bit 0 of AC2 is set to 0.

Next, the current interrupt priority mask is pushed on the stack. The contents of the second word of DCT are logically OR'd with the current interrupt priority mask and the result is placed in both AC0 and memory location 5. This constructs the new interrupt priority mask and places it in AC0 and the save location for the mask. A DOBS 0,CPU instruction is now performed. This is a MASK OUT instruction that also enables the interrupt system.

After a new interrupt priority mask is established and the interrupt system enabled, control is transferred to the device handler by placing bits 1-15 of the first word of the DCT in the program counter.


START OF VCT INSTRUCTION
FETCH THE SECOND WORD OF THE VCT INSTRUCTION. BIT 0 IS THE STACK CHANGE BIT. BITS 1-15 CONTAIN THE ADDRESS OF THE BEGINNING OF THE VECTOR TABLE
PERFORM INTA
ADD THE CODE RETURNED FROM INTA TO THE ADDRESS OF THE VECTOR TABLE AND FETCH THE WORD AT THAT LOCATION. BIT 0 IS THE "DIRECT BIT"
DIRECT BIT = 0?
YES
NO
BITS 1-15 OF THE FETCHED VECTOR TABLE ENTRY CONTAINS THE ADDRESS OF THE DCT
BITS 1-15 OF THE FETCHED VECTOR TABLE ENTRY CONTAIN THE ADDRESS OF THE DEVICE INTERRUPT ROUTINE.
NO
STACK CHANGE BIT = 1?
YES
TRANSFER CONTROL TO THE DEVICE INTERRUPT ROUTINE BY PLACING BITS 1-15 OF THE FETCHED VECTOR TABLE ENTRY IN THE PROGRAM COUNTER
SAVE LOCATIONS $40-43_8$
PLACE CONTENTS OF LOCATION 4 IN STACK POINTER. PLACE CONTENTS OF LOCATION 6 IN STACK LIMIT. PLACE CONTENTS OF LOCATION 7 IN STACK FAULT. NOTE: FRAME POINTER IS DESTROYED AND THE CONTENTS ARE UNPREDICTABLE
PUSH OLD CONTENTS OF LOCATIONS $20_{16}23_{16}$ (40_8-43_8)
A
DG-01133
A
A
FETCH THE FIRST WORD OF THE DCT. BIT 0 IS THE "PUSH BIT". BITS 1-15 CONTAIN THE ADDRESS OF THE DEVICE INTERRUPT ROUTINE.
NO
PUSH BIT = 1?
YES
PUSH STANDARD RETURN BLOCK. BITS 1-15 OF LAST WORD PUSHED CONTAIN BITS 1-15 OF PHYSICAL LOCATION 0.
PLACE THE ADDRESS OF THE DCT IN AC2.
PUSH THE CURRENT INTERRUPT MASK (LOCATION 5) ONTO THE STACK.
PLACE THE LOGICAL OR OF THE CURRENT INTERRUPT MASK AND THE SECOND WORD OF THE DCT IN AC0.
PLACE THE CONTENTS OF AC0 IN THE CURRENT INTERRUPT MASK (LOCATION 5).
DO A MASK OUT FROM AC0 AND ENABLE INTERRUPTS (DOBS 0,CPU).
PLACE ADDRESS OF DEVICE INTERRUPT ROUTINE IN PROGRAM COUNTER.
STACK OVERFLOW ?
YES
NO
CONTINUE SEQUENTIAL OPERATION WITH THE WORD ADDRESSED BY THE PROGRAM COUNTER
TRANSFER CONTROL TO STACK FAULT ROUTINE
END OF VCT INSTRUCTION

Use of the Vector Instruction

The VECTOR ON INTERRUPTING DEVICE CODE instruction is an extremely powerful instruction. Because of the impact of interrupt latency on overall system performance, and the impact of the VECTOR instruction on interrupt latency, this instruction should be well understood before it is used.

The VECTOR instruction can operate in any one of five modes. These modes are called mode A, mode B, mode C, mode D, and mode E. In general, as one goes through the modes, from A to E, the instruction performs more work, giving the user more power, but also takes more time to execute.

For all modes, the VECTOR instruction uses bits 1-15 of the second word to address the vector table. An INTERRUPT ACKNOWLEDGE instruction is performed and the device code received is added to the address of the vector table and the word at that location is fetched. At this point, the mode selection process begins.

Which mode is used for execution is a function of the direct bit in the vector table entry, the stack change bit in the second word of the VECTOR instruction and the push bit in the first word of the DCT. The table below gives the relationship.

DIRECT	STACK CHANGE	PUSH	MODE
0	X	X	A
1	0	0	B
1	0	1	C
1	1	0	D
1	1	1	E

Note: X = Don't care

For mode A, the state of the stack change and push bits don't matter because they are never checked.

The uses of the five modes are described below.

Mode A is used when no time can be wasted in getting to the interrupt handler for a device. A device requiring mode A service would typically be a non-buffered device with a very small latency time. Alternatively, a real time process that must receive control immediately after an event could be serviced using mode A. The programmer pays for the speed realized through mode A by giving up the state saving and priority masking features of the other modes.

Modes B, C, D, and E are used to implement a priority interrupt structure. They all build a new priority mask and save the old priority mask before issuing a MASK OUT instruction that enables the interrupt system. These modes differ in the amount of time and work that they devote to saving the state of the machine.

In a priority system, there are typically two types of processes: those that operate at "base" level, and those that do not. Base level is defined as operating with all levels of interrupt enabled and no interrupt processing in progress. Non-base level is defined as operating with some interrupt processing in progress. In general, those processes that operate at base level are user problem programs. Those processes that operate at non-base level are the various interrupt handlers in the system.

One of the first things that the supervisor program should do when it receives an interrupt while a process is operating at base level is to change the stack environment. Two reasons lead to this conclusion. The supervisor has no control over whether or not the user has defined a stack by placing meaningful information in the stack control words. Additionally, even if the user has initialized a stack, the supervisor has no control over the size of the stack. If the user has defined a stack, but is very close to his stack limit, it would not be acceptable for a supervisor interrupt routine to fill the user's stack to overflowing. By using either mode D or E, the VECTOR instruction will change the stack environment and initialize a stack over which the supervisor has full control. At the same time, the VECTOR instruction will save the stack environment of the user so that it may be restored before control is returned to the user.

If an interrupt handler is already processing when another interrupt is received, then the stack environment has already been changed by the interrupt that occurred at base level and should not be changed again. For interrupts that occur at non-base level, modes B and C of the VECTOR instruction can be used.

The difference between modes D and E is the same as the difference between modes B and C: modes B and D do not push a return block onto the stack.

While this saves a little bit of time over modes C and E, it makes returning control to the interrupted program somewhat more complicated.

All modes of the VECTOR instruction can be combined in one vector table. Devices that require mode A service will have bit 0 set to 0 in their vector table entry. The other devices will have bit 0 set to 1 in their vector table entries, and control their modes of service by the setting of the push bit in their DCT's.

This page intentionally left blank

DATA CHANNEL FACILITY

INTRODUCTION

Peripherals which need to transfer large blocks of data quickly generally accomplish their data transfers via the data channel facility. The actual data channel transfers do not disturb the state of the processor since the data is transferred directly between registers in the controller and memory. This means that the amount of program overhead in the form of executing I/O instructions and loading or storing data is greatly reduced. The time required for program execution is lengthened however, since the processor pauses, as soon as it is able, each time a word is to be transferred; the transfer then occurs and the processor continues. The program need only set up the peripheral for the transfer and can then perform other, unrelated tasks.

Features

The data channel facilities in the original NOVA, NOVA 1200 series, and the ECLIPSE computers all provide a single speed for data channel operation. The SUPERNOVA® series, NOVA 800 series, and NOVA 2 series computers all can operate the data channel at two different speeds: standard and high speed. In addition to merely transferring data, certain arithmetic operations can be performed by the data channel in some computers. All the NOVA line computers can have the contents of any memory location incremented by 1 each time a controller requests that operation. The NOVA and SUPERNOVA computers also allow a controller to add a word to the previous contents of any memory location.

In both types of arithmetic operation, the computer sends the results back to the controller and, if the operation increased the contents of the memory location to more than 2^{16}-1, it sends an overflow signal.

The data channel allows many peripherals to be active at the same time, providing access to memory to individual controllers on demand. Peripherals which use the data channel operate under a priority structure imposed on them by the channel. In cases where more than one controller requests access to the data channel at the same time, priority is given to that controller which is closest to the processor on the I/O bus.

A table in Appendix D includes the maximum transfer rates for all combinations of channel speed and type of transfer.

CONTROLLER STRUCTURE

Understanding the operation of the data channel requires a knowledge of the structure of the controllers which use it. The controllers usually contain the normal Busy and Done flags, status, control, and information registers, and the program interrupt components. Additional components are added to handle the functions necessary to operate the data channel. Some of these components, generally available to the program, are in the form of additional control and status registers.

The two main registers usually added are the Word Counter and the Memory Address Counter. The Word Counter is used by the program to specify the size of the block of data to be transferred (number of words). The Memory Address Counter is used to specify the address in memory which is used in the data transfer.

Word Counter

The Word Counter is loaded, by the program, with the two's complement of the number of words in the block. Each time a word is transferred, the controller automatically increments the counter by 1. When the counter overflows, the controller terminates data channel transfers.

The size of the Word Counter varies from peripheral to peripheral, depending on the block size associated with the peripheral. Typical sizes of the Word Counter are 12 and 16 bits, allowing for up to 4096-word blocks and 65,536-word blocks, respectively. Although the Word Counter specifies the negative of the desired block size, the most significant bit of the register need not be a 1--it is not a sign bit for the number. No sign bit is necessary because the word count is treated as negative by the controller, by virtue of being incremented instead of decremented. Thus, a word count of 0 is valid; in fact, it specifies the largest possible block size. The table below further illustrates the correspondence between the desired word count and the value which must be loaded into a 12-bit or 16-bit Word Counter.

(negative) word count (decimal)	16-bit value (octal)	12-bit value (octal)
-1	177777	7777
-2	177776	7776
-8	177770	7770
-100	177634	7634
-2047	174001	4001
-2048	174000	4000
-2049	173777	3777
-4095	170001	0001
-4096	170000	0000
-4097	167777	
-8192	160000	
-32768	100000	
-65535	000001	
-65536	000000	

Memory Address Counter

The Memory Address Counter always contains the address in memory which is to be used by the controller for the next data transfer. It is loaded, by the program, with the address of the first word in the block to be transferred. During each transfer, the controller increments the Memory Address Counter by 1. Therefore, successive transfers are to or from consecutive memory locations.

TRANSFER SEQUENCE

The actual data channel transfer sequence is a two-way communication between processor and controller and proceeds as follows. When a peripheral has a word of data ready to be transferred to memory or wants to receive a word from memory, it issues a data channel request to the processor. The processor pauses as soon as it is able, and begins the data channel cycle by acknowledging the peripheral's data channel request. The acknowledgment signal dismisses the peripheral's data channel request and causes the peripheral to send back to the processor the address of the memory location involved in the transfer. Following the receipt of the address, the data itself is transferred in the appropriate direction.

At the completion of each data transfer the processor/controller interaction is over. The controller carries out any tasks necessary to complete the data transfer, such as transferring the data to the device itself for an output operation. The processor starts another data channel transfer if any data channel requests are pending, starts a program interrupt if one is being requested and there are no data channel requests, or resumes program execution.

The controller increments both the Memory Address Counter and the Word Counter during the transfer. If the word count becomes 0, the controller terminates further transfers, sets the Busy flag to 0, the Done flag to 1, and initiates a program interrupt request. If the Word Counter has not yet overflowed, the peripheral continues its operation, issuing another data channel request when it is ready for the next transfer.

Processor Pauses

The processor can pause for a data channel transfer only at certain, well-defined times, depending on the model of processor and, in the SUPERNOVA computer, on the channel speed used by the peripheral requesting the transfer. For the NOVA, NOVA 1200 series, and the standard channel on SUPERNOVA computers, data channel transfers, like program interrupts, can only occur between instructions. High-speed data channel requests on the SUPERNOVA computer, and all requests on the NOVA 800 series, NOVA 2 series, and ECLIPSE computers can break into program execution at certain other points in most instructions (I/O instructions are included in those that cannot be broken into.)

Priorities

In terms of priorities, program execution has priority over the data channel except at certain points in the processor's operation, at which times the data channel has absolute priority (over not only normal program execution but also over any pending program interrupt requests). At these certain points, the processor will handle all existing data channel requests, including those which are generated while data channel transfers are in progress, before starting a program interrupt or resuming normal instruction execution. Thus, if data channel requests are being generated by a number of peripherals as fast as or faster than the processor can handle them, all processing time will be spent handling data channel transfers, and program execution will stop until all the data channel transfers are made. However, when the data channel is being used at less than its maximum rate, processing time is shared between the data channel, which receives as much as it needs, and the program, which uses the rest.

When the processor pauses to honor a data channel request and more than one peripheral is requesting a data channel transfer, priority is given to the peripheral which is closest to the processor on the I/O bus. Since all peripherals operating with the high-speed data channel must be grouped together at the beginning of the bus, all requests from high-speed controllers will be honored before any from those which operate at standard speed. To use the high-speed data channel, the controller for a peripheral must be mounted inside the mainframe of the computer and must be designed to operate within the high-speed data channel time constraints. (Refer to the Interface Manual, DGC 015-000031.) A computer that has the two-speed capability is shipped with the high speed enabled for all controllers mounted inside the mainframe. (Controllers in an expansion chassis are constrained to operate at standard speed.)

PROGRAMMING

Programming a peripheral for a data channel block transfer typically involves the following steps:

1. The peripheral's status is checked, usually by testing the Busy flag and/or reading a status word and checking one or more error or ready bits. If an error has occurred, the program should take appropriate action. If no error has occurred but the peripheral is not yet ready, the program should wait for the peripheral to complete its operation. When the peripheral is ready, the program may proceed.
2. The data block in the device is located. This usually involves giving a peripheral "address" by specifying a unit number, channel number, sector number, or the like.
3. The data block in memory is located by loading the Memory Address Counter with the address of the first word of the block.
4. The size of the data blocks is specified by loading the proper value into the Word Counter.
5. The type of transfer is specified and the operation is initiated. If the peripheral is capable of several different operations, specifying the type of transfer usually involves loading a control register in the controller. The operation itself is usually initiated by one of the I/O control functions: Start, or Pulse.

Setting up and initiating the data channel operation is the major part of programming a data channel block transfer. However, if any errors could have occurred during the operation, the program should check for these errors when the operation is complete and take appropriate action if one or more have occurred.

This page intentionally left blank

TIMING

INTRODUCTION

On large systems which depend heavily on input/output, both the direct program control and data channel facilities can be badly overloaded. This overloading means that certain peripherals are seriously compromised because they lose data or perform poorly, since the system cannot respond to them in time.

This section explains how a system can be overloaded and what steps can be taken to minimize the detrimental effects.

DIRECT PROGRAM CONTROL

Nearly all peripherals operating under direct program control request program service by setting their Done flags to 1. Whether the CPU determines that the Done flag is set to 1 by repeatedly checking it or by responding to interrupt requests, there may be a significant delay between the time the peripheral requests program service and when the CPU carries out that service. This delay is called "access time".

When the program interrupt facility is not used, access time has two components which can be calculated from the tables in Appendix D.

1. The interval between the time the Done flag is set to 1 by the peripheral and the time the flag is checked by the CPU.
2. The time required by the peripheral service routine to transfer data to/from the peripheral and set the Done flag to 0 (by idling the peripheral or instructing it to begin a new operation.

The first component can be diminished by performing frequent checks on the Done flag; the second can be diminished by writing an efficient peripheral service routine.

When the program interrupt facility is used, the access time has at least four components:

1. The time from the setting of the Done flag to 1, to the end of the instruction being executed by the CPU.
2. The time the interrupt facility needs to store the program counter in location 0 and simulate a JMP @1 instruction.
3. The time required by the interrupt handler to identify the peripheral and transfer control to the service routine.
4. The time required by the service routine to transfer data to/from the peripheral and set the Done flag to 0.

The access time may be extended by three other components:

5. The time when CPU operation is suspended because data channel transfers are in progress (see following section).
6. The time during which the CPU does not respond to the peripheral's interrupt request because the interrupt system is disabled. (For example, during the servicing of an interrupt from another peripheral.)
7. The time the peripheral's Interrupt Disable flag is set to 1 during the servicing of an interrupt of a higher level peripheral.

The first component is determined by the longest non-interruptable instruction that the CPU can execute. On the NOVA line computers, this is usually a few microseconds (unless long indirect address chains are used in several processors); on the ECLIPSE computer it can be considerably longer due to the presence of the WCS feature which allows the programmer to code long instructions which do not allow program interrupts to occur during their execution.

The second component is also machine dependent; in general it is approximately three times as long as a memory reference. The third, fourth, sixth and seventh components are determined by software and account for the bulk of the access time. The fifth component is determined by the nature and the number of the data channel devices operating in the system.

Access time is important because a peripheral that must wait too long for program service from the CPU may suffer from degraded performance. The longest allowable delay between the time when a peripheral sets its Done flag to 1 and the time when the CPU transfers data to/from that peripheral and sets the Done flag to 0 is called the "maximum access time" of the peripheral. When the actual access time for a peripheral exceeds the maximum access time, the specific effects depend on the device in question. In the worst case, data may be incorrectly read or written. The maximum allowable access times for each peripheral can be found under Timing in the section devoted to that peripheral.

A peripheral service routine must usually perform certain computations (updating pointers to buffers, byte counters, etc.), but rarely are these computations so complex that they cannot be accomplished within the constraints of the maximum allowable access time. However, if several peripherals are competing for service at the same time, it may be necessary to jeopardize the performance of some of them be deferring their requests for program service until the CPU has serviced the higher priority requests. For this reason, all DGC computers incorporate the priority interrupt facility described earlier.

The object of the priority interrupt facility is to minimize the loss of important data. In order for the programmer to achieve this end, the assignment of the software priority levels should be made in the light of the following considerations:

1. the maximum allowable access time for each peripheral,
2. the result of exceeding the maximum allowable access time for each peripheral (slowdown or data loss), and
3. the cost of losing data.

DATA CHANNEL CONTROL

Problems with time constraints can be encountered when transferring data via the data channel. When a peripheral needs data channel service, it makes a data channel request. However, the CPU can only allow data channel peripherals to access memory at certain predefined times. (At such times, it is said that data channel breaks are enabled.) In addition, there may be more than one peripheral waiting to access memory at any one time. Consequently, there may be a significant delay between the time when a peripheral requests access to memory and the time when the transfer actually occurs. This delay is called data channel latency and has the following components:

1. the time between the peripheral's request for memory access and the next data channel break, and
2. the time required to complete data channel transfers to/from higher priority (closer) peripherals that are also requesting memory access.

The length of the first component depends on the design of the CPL. In the NOVA, NOVA 1200 series, and SUPERNOVA (standard-speed) computers, data channel breaks are enabled only between instructions so that long instructions (MUL, DIV) and long indirect address chains can have a significant effect on data channel latency. In the NOVA 800 series, NOVA 2 series, ECLIPSE and SUPERNOVA (high-speed) computers, data channel breaks may be enabled during most instructions (but not during I/O instructions), so that data channel latency is reduced.

The length of the second component depends on the number of data channel peripherals operating in the system at a higher priority and the frequency of their use.

Most peripherals using the data channel control operate under fixed time constraints. Disc drives and magnetic tape transports are typical data channel peripherals. In each of these devices, a magnetic medium moves past a read or write head at constant velocity. If data is not read or written at the correct instant, the data will be transferred to or from the wrong place on the magnetic medium. Consequently, on input, such devices must be allowed to write a word into memory before the next word is assembled by the controller, and on output, the controller must be able to read a word from memory before the surface is positioned under the write head. In either case, if the data channel latency is too long, data cannot be properly transferred. Most peripherals operating under data channel control set an error flag when this happens, so the service routine can take appropriate action to recover from the error, if possible.

The maximum allowable data channel latency of a peripheral is the maximum time the peripheral can wait for a data channel transfer. The range of times is from a few microseconds to several hundred microseconds. At the time the system is configured, data channel priorities should be assigned to peripherals on the basis of the following considerations:

1. The maximum allowable data channel latency of the peripheral. A peripheral with a short allowable latency usually should receive a higher priority than one with a long allowable latency.
2. The recovery time of a peripheral (i.e., how long before it can repeat a transfer that failed because of excessive data channel latency) if the peripheral can recover.
3. The cost of losing data from the peripheral if the peripheral cannot recover.

If data channel latency seems to be a problem in a system, latency might be improved by changing the coding in the programs; less frequent use of long instructions and lengthy indirect chains in the NOVA and 1200 series computers, and less frequent use of I/O instructions in the SUPERNOVA, 800 series, and ECLIPSE computers. In addition, there is an upper limit on the number of data channel transfers/second that a computer can support. In cases where this limit is exceeded, the only solution is to reduce the number of peripherals using the data channel at the same time.

High data channel use reduces the speed of program execution since the processor pauses for each transfer. This may adversely affect the CPU's capacity to respond to interrupts and service those peripherals operating under direct program control.

This page intentionally left blank

PROGRAMMING EXAMPLES

INTRODUCTION

The four programs in this section illustrate various techniques of I/O programming. Some of the techniques used are: status checking; servicing of basic peripherals (Teletype, paper tape punch and reader); interrupt handling, including saving and restoring the processor state, with and without stacks, and dispatching to the proper peripheral service routine; data channel programming; priority interrupt handling; and power fail/auto restart programming. The programs are complete; if desired, each one can be assembled and run on any DGC computer system with the necessary peripherals.

The programs become successively more complex in order to illustrate various programming methods and to show the costs and benefits of the use of interrupts. As the complexity increases, so does the efficiency of peripheral servicing. Example One uses no interrupts in servicing Teletype and punch. Example Two performs the same service more efficiently by using the interrupt facility. In Example Three, a priority interrupt structure is introduced to provide the most efficient service for an expanded set of peripherals which includes a paper tape reader and magnetic tape unit. Example Four, for use on the ECLIPSE computer, utilizes the VECTOR instruction and stack facilities to perform the bulk of the tasks performed by the interrupt handler. However, the Teletype and punch service in Examples Three and Four remains functionally the same as before, so that the programming styles represented by the example programs can easily be compared.

The basic idea of each program is the same. Program computation is simulated by a short loop of arithmetic instructions. A counter in memory (LOC) is incremented on each pass through the loop in order to provide a recognizable pattern of blinking console lights. Periodically, these calculations are suspended in order to perform I/O, at the completion of which the calculations are resumed exactly where they were left off.

The operation of the programs is functionally identical with respect to Teletype and punch I/O. When a key is typed, the calculations are suspended and the character is read in, stored in a buffer, and "echoed" back to the Teletype. When the line is terminated, either by a carriage return or by the 72nd character, the entire line is output to the punch. (In order to extend the punching time for a single line, each character is punched eight times.) The buffer is then reinitialized and the program waits for more Teletype input as it continues its calculations.

EXAMPLE ONE

Example One is written without using interrupts. (The interrupt system is disabled by the initial IORST instruction.) The status of the Teletype keyboard (device TTI) is checked periodically (roughly every half a second). When a character has been typed and is waiting to be read (Done flag set to 1), the calculations are frozen and saved (at IOSR) and the character is read in (READC), stored in the buffer (STORE), and echoed back to the Teletype. If this character does not terminate the line, the frozen state of the calculations is restored (RSTOR) and the calculations are resumed. When a carriage return is read, it is echoed and placed into the buffer as usual, after which a line feed is added to the buffer and echoed. If 72 characters are input with no carriage return, both a carriage return and a line feed are sent to the Teletype and put into the buffer. In either case the program then loops through the buffer, punching out each character in the line eight times (NRPT=8). (A null character is output first to provide eight frames of blank leader on the tape.) When the full line has been punched this way, the program reinitializes the character buffer and resumes the calculations.

When Example One is actually run, it becomes obvious that this is not an efficient way to handle this type of I/O. The Teletype seems sluggish; the infrequent checking of the Teletype status introduces a delay before a character is echoed. In addition, while a line is being punched, no characters are recognized from the Teletype and all calculations stop (the console lights stop blinking). The obvious way to reduce this poor performance is to use the interrupt facility, as Example Two does.

EXAMPLE ONE

```
        .LOC  20
        BUFF-1              ;AUTO-INCREMENTING LOCATION USED AS
                            ; INDEX INTO CHARACTER BUFFER

        .LOC  400

;       "MAIN" PROGRAM

START:  IORST               ;CLEAR I/O DEVICES, TURN INTERRUPTS OFF

CALC:   ADD   0,1           ;SIMULATE USEFUL COMPUTATION
        SUB   1,2
        INC   2,3
        NEG   3,0
        ISZ   LOC
        JMP   CALC
        SKPDN TTI           ;CHECK TTI STATUS PERIODICALLY
        JMP   CALC          ;CONTINUE COMPUTATION
        JMP   IOSR          ;IF CHARACTER IS WAITING GO GET IT

LOC:    0


;       INPUT/OUTPUT SERVICE ROUTINE

IOSR:   STA   0,SAVE0       ;SAVE ACCUMULATORS THAT WILL BE USED
        STA   3,SAVE3
        MOVL  0,0           ;SAVE CARRY
        STA   0,SAVEC

READC:  DIAC  0,TTI         ;READ THE CHARACTER AND CLEAR DONE
        LDA   3,C177        ;DROP PARITY BIT
        AND   3,0
STORE:  STA   0,@20         ;STORE CHARACTER IN BUFF
        SKPBZ TTO           ;TTO READY?
        JMP   .-1           ;NO, TRY AGAIN
        DOAS  0,TTO         ;YES, ECHO CHARACTER
        LDA   3,CR          ;IS CHARACTER A CARRIAGE RETURN?
        SUB#  3,0,SNR
        JMP   CRET          ;YES, GO PROCESS IT
        DSZ   CHCNT         ;NO, DECREMENT CHCNT TO CHECK FOR LINE OVERFLOW
        JMP   RSTOR         ;NO OVERFLOW, EXIT
        MOV   3,0           ;LINE IS FULL, LOAD A CR AND
        JMP   STORE         ; JUMP BACK TO INSERT IT INTO BUFF
```

EXAMPLE ONE (Continued)

```
CRET:   LDA   0,LF         ;ADD LINE FEED TO BUFF
        STA   0,@20
        SKPBZ TTO          ;WAIT UNTIL  TTO READY
        JMP   .-1
        DOAS  0,TTO        ;GIVE LINE FEED
        LDA   0,20         ;CALCULATE NUMBER OF CHARACTERS IN BUFF
        LDA   3,ABUFF
        SUB   3,0
        STA   0,PCCNT      ;INITIALIZE PUNCH CHARACTER COUNTER
        STA   3,20         ;RESET BUFFER INDEX TO BEGINNING OF BUFF
        SUB   0,0,SKP      ;FIRST PUT OUT 8 FRAMES OF BLANK LEADER
NEWC:   LDA   0,@20        ;GET NEW CHARACTER
SAMEC:  SKPBZ PTP          ;WAIT UNTIL  PUNCH READY
        JMP   .-1
        DOAS  0,PTP        ;OUTPUT CHARACTER TO PUNCH
        DSZ   PRCNT        ;DECREMENT PUNCH REPEAT COUNTER
        JMP   SAMEC        ;GO PUNCH SAME CHARACTER AGAIN
        LDA   0,NRPT       ;REINITIALIZE PUNCH REPEAT COUNTER
        STA   0,PRCNT
        DSZ   PCCNT        ;DECREMENT PUNCH CHARACTER COUNTER
        JMP   NEWC         ;START PUNCHING NEW CHARACTER
        LDA   0,MAX        ;ALL DONE, REINITIALIZE CHARACTER BUFFER:
        STA   0,CHCNT      ; INITIALIZE CHARACTER COUNTER
        LDA   0,ABUFF      ; INITIALIZE CHARACTER BUFFER INDEX
        STA   0,20

RSTOR:  LDA   0,SAVEC      ;RESTORE CARRY AND ACCUMULATORS
        MOVR  0,0
        LDA   0,SAVE0
        LDA   3,SAVE3
        JMP   CALC         ;RETURN AND RESUME COMPUTATION

C177:   177                ;MASK FOR DELETING PARITY BIT
CR:     15                 ;CARRIAGE RETURN
LF:     12                 ;LINE FEED

ABUFF:  BUFF-1             ;ADDRESS OF CHARACTER BUFFER - 1
MAX:    110                ;MAXIMUM OF 72 CHARACTERS INPUT PER LINE
CHCNT:  110                ;CHARACTER COUNTER (COUNTS DOWN FROM MAX)

PCCNT:  0                  ;PUNCH CHARACTER COUNTER
PRCNT:  10                 ;PUNCH REPEAT COUNTER
NRPT:   10                 ;NUMBER OF TIMES TO PUNCH EACH CHARACTER

SAVE0:  0                  ;ACCUMULATOR SAVE LOCATIONS
SAVE3:  0
SAVEC:  0                  ;CARRY SAVE LOCATION

BUFF:   .BLK  112          ;CHARACTER BUFFER - ALLOW FOR 72 CHARS + CR + LF

        .END  START
```

EXAMPLE TWO

Example Two performs the same input/output operations that Example One does, but with its use of the interrupt facility it functions much more efficiently. Two major benefits of the interrupt are illustrated:

1. The program need not check the TTI status while it is performing its calculations--the calculations are interrupted automatically when there is I/O to be done. This eliminates the sluggishness encountered in Example One: in Example Two the response to typing a character appears to be immediate.
2. In the periods during which the computer is waiting for an I/O (in this case the punch) to complete an operation before starting it on another operation, a program which uses interrupts can perform other computation. When Example Two is actually run, the console lights continue to show calculations being performed while the punch is running. Note, however, that while the punching is being done, TTI interrupts must be ignored. Otherwise, a carriage return given while punching was going on would terminate the line being punched and start the new one. A program check is employed in Example Two to avoid this problem. Right after a character is read in (READC), if a line is being punched (PCCNT>0), the character is ignored and the interrupt is dismissed immediately. In Example Three, use of a priority interrupt scheme eliminates this problem, so that characters can be accepted from the Teletype during punching.

A polling routine is used in Example Two to determine which peripheral caused the interrupt. The I/O SKIP instructions (beginning at IHAND) are ordered so that the higher-priority devices are interrogated first. Although the order is not critical for such slow devices as are used here, it is generally good practice to check the higher-priority peripherals first.

Three other things should be noted. The first instruction (at START) is now DICS 0,CPU. This resets all peripherals as does an IORST, but it also enables, rather than disables, the interrupt facility. Secondly, Teletype output (TTO) interrupts are unconditionally ignored by merely clearing the Done flag and returning. Thirdly, for computers with the power monitor and auto-restart feature, a power fail service routine has been written. In Example Two this routine (PFLSR) merely saves the state of the processor and halts. A more involved routine, which provides auto-restart capability, is included in Example Three.

EXAMPLE TWO

```
        .LOC   0
        0                 ;PC STORED HERE ON INTERRUPT
        IHAND             ;INTERRUPT HANDLER ADDRESS

        .LOC   20
        BUFF-1            ;AUTO-INCREMENTING LOCATION USED AS
                          ; INDEX INTO CHARACTER BUFFER

        .LOC   400

;       "MAIN" PROGRAM

START:  DICS   0,CPU      ;IORST AND INTEN

CALC:   ADD    0,1        ;SIMULATE USEFUL COMPUTATION
        SUB    1,2
        INC    2,3
        NEG    3,0
        ISZ    LOC
        JMP    CALC
        JMP    CALC       ;NO NEED TO CHECK HERE FOR TTI DONE

LOC:    0


;       INTERRUPT HANDLER

IHAND:  SKPDZ  CPU        ;POLL DEVICES TO LOCATE SOURCE OF INTERRUPT
        JMP    PFLSR      ;POWER FAIL INTERRUPT
        SKPDZ  PTP
```

EXAMPLE TWO (Continued)

```
        JMP    PTPSR       ;PAPER TAPE PUNCH INTERRUPT
        SKPDZ  TTI
        JMP    TTISR       ;TELETYPE INPUT INTERRUPT
        SKPDN  TTO
        HALT               ;SPURIOUS INTERRUPT
        NIOC   TTO         ;TELETYPE OUTPUT, IGNORE INTERRUPT
        INTEN              ;ENABLE INTERRUPTS
        JMP    @0          ;RETURN TO INTERRUPTED PROGRAM


;       POWER FAIL SERVICE ROUTINE

PFLSR:  STA    0,SAVE0     ;POWER FAILURE, SAVE ACCUMULATORS AND CARRY
        STA    1,SAVE1
        STA    2,SAVE2
        STA    3,SAVE3
        MOVL   0,0
        STA    0,SAVEC
        HALT               ;WAIT FOR POWER TO GO DOWN


SAVE0:  0                  ;ACCUMULATOR SAVE LOCATIONS
SAVE1:  0
SAVE2:  0
SAVE3:  0
SAVEC:  0                  ;CARRY SAVE LOCATION


;       TELETYPE INPUT SERVICE ROUTINE

TTISR:  STA    0,SAVE0     ;SAVE ACCUMULATORS THAT WILL BE USED
        STA    3,SAVE3
        MOVL   0,0         ;SAVE CARRY
        STA    0,SAVEC

READC:  DIAC   0,TTI       ;READ THE CHARACTER AND CLEAR DONE
        LDA    3,C177      ;DROP PARITY BIT
        AND    3,0
        LDA    3,PCCNT     ;IS PREVIOUS LINE BEING PUNCHED?
        MOV#   3,3,SZR
        JMP    TTIDS       ;IF YES, THEN IGNORE THIS CHARACTER
STORE:  STA    0,@20       ;STORE CHARACTER IN BUFF
        SKPBZ  TTO         ;TTO READY?
        JMP    .-1         ;NO, TRY AGAIN
        DOAS   0,TTO       ;YES, ECHO CHARACTER
        LDA    3,CR        ;IS CHARACTER A CARRIAGE RETURN?
        SUB#   3,0,SNR
        JMP    CRET        ;YES, GO PROCESS IT
        DSZ    CHCNT       ;NO, DECREMENT CHCNT TO CHECK FOR LINE OVERFLOW
        JMP    TTIDS       ;NO OVERFLOW, GO DISMISS INTERRUPT
        MOV    3,0         ;LINE IS FULL, LOAD A CR AND
        JMP    STORE       ; JUMP BACK TO INSERT IT INTO BUFF
CRET:   LDA    0,LF        ;ADD LINE FEED TO BUFF
        STA    0,@20
        SKPBZ  TTO         ;WAIT UNTIL TTO READY
        JMP    .-1
        DOAS   0,TTO       ;GIVE LINE FEED
        LDA    0,20        ;CALCULATE NUMBER OF CHARACTERS IN BUFF
        LDA    3,ABUFF
        SUB    3,0
        STA    0,PCCNT     ;INITIALIZE PUNCH CHARACTER COUNTER
        STA    3,20        ;RESET BUFFER INDEX TO BEGINNING OF BUFF
        SUB    0,0         ;FIRST PUT OUT 8 FRAMES OF BLANK LEADER
        DOAS   0,PTP
        STA    0,CCHAR     ;SAVE NULL AS CURRENT CHARACTER

TTIDS:  LDA    0,SAVEC     ;TTI INTERRUPT DISMISSAL
        MOVR   0,0         ;RESTORE CARRY AND ACCUMULATORS
```

EXAMPLE TWO (Continued)

```
        LDA     0,SAVE0
        LDA     3,SAVE3
        INTEN                   ;ENABLE INTERRUPTS
        JMP     @0              ;RETURN TO INTERRUPTED PROGRAM

C177:   177                     ;MASK FOR DELETING PARITY BIT
CR:     15                      ;CARRIAGE RETURN
LF:     12                      ;LINE FEED

ABUFF:  BUFF-1                  ;ADDRESS OF CHARACTER BUFFER - 1
MAX:    110                     ;MAXIMUM OF 72 CHARACTERS INPUT PER LINE
CHCNT:  110                     ;CHARACTER COUNTER (COUNTS DOWN FROM MAX)

PCCNT:  0                       ;PUNCH CHARACTER COUNTER
PRCNT:  10                      ;PUNCH REPEAT COUNTER
NRPT:   10                      ;NUMBER OF TIMES TO PUNCH EACH CHARACTER
CCHAR:  0                       ;CURRENT CHARACTER SAVE LOCATION


;       PAPER TAPE PUNCH SERVICE ROUTINE

PTPSR:  STA     0,SAVE0         ;SAVE ACCUMULATORS THAT WILL BE USED
        STA     3,SAVE3         ;(CARRY WILL NOT BE CHANGED)

GCHAR:  LDA     0,CCHAR         ;GET CURRENT CHARACTER
        DSZ     PRCNT           ;DECREMENT PUNCH REPEAT COUNTER
        JMP     SAMEC           ;GO PUNCH SAME CHARACTER AGAIN
        LDA     0,NRPT          ;REINITIALIZE PUNCH REPEAT COUNTER
        STA     0,PRCNT
        DSZ     PCCNT           ;DECREMENT PUNCH CHARACTER COUNTER
        JMP     NEWC            ;GO PUNCH NEW CHARACTER
        NIOC    PTP             ;ALL DONE, CLEAR PUNCH
        LDA     0,MAX           ;REINITIALIZE CHARACTER BUFFER:
        STA     0,CHCNT         ; INITIALIZE CHARACTER COUNTER
        LDA     0,ABUFF         ; INITIALIZE CHARACTER BUFFER INDEX
        STA     0,20
        JMP     PTPDS           ;GO DISMISS INTERRUPT
NEWC:   LDA     0,@20           ;GET NEW CHARACTER
        STA     0,CCHAR         ;UPDATE CURRENT CHARACTER SAVE
SAMEC:  DOAS    0,PTP           ;OUTPUT CHARACTER TO PUNCH

PTPDS:  LDA     0,SAVE0         ;PTP INTERRUPT DISMISSAL
        LDA     3,SAVE3         ;RESTORE ACCUMULATORS
        INTEN                   ;ENABLE INTERRUPTS
        JMP     @0              ;RETURN TO INTERRUPTED PROGRAM


BUFF:   .BLK    112             ;CHARACTER BUFFER - ALLOW FOR 72 CHARS + CR + LF

        .END    START
```

EXAMPLE THREE

Example Three creates a full priority interrupt structure and provides even more efficient I/O processing. Two peripherals are added to the program: the high-speed paper tape reader, as a peripheral which requires service quickly enough that it could benefit from such a priority interrupt structure, and a magnetic tape drive, as a peripheral which uses the data channel. In addition to the Teletype and punch I/O of the previous two examples, Example Three allows a block of information to be read in through the high-speed reader, stored in a second buffer, and written out to magnetic tape. This sequence is initiated by typing control-R at any point in a Teletype input line. Doing so will store the control-R in the buffer but echo it by ringing the Teletype bell.

When the completed line is punched, any control-R's in the line are punched only once (instead of the usual eight times), and it starts the paper tape reader. For each control-R, one block of paper tape (delimited by one or more null frames) is read. After a block is read, 100-word sections of the block are written to magnetic tape via the data channel. (A block on paper tape produced by this program on the punch can be as long as 74 x 8 = 592 frames, equivalent to several magnetic tape records of the arbitrary length 100.) When all the writing is done, the paper tape reader is started again if there were additional control-R's input.

Priority Structure

The interrupt priority masks are set up to create the following priority structure among active peripherals, from highest priority to lowest:

paper tape reader
magnetic tape
paper tape punch
Teletype input
Teletype output

The paper tape reader is given highest priority because it has the shortest allowable access time. In order to keep the reader operating at maximum speed, response to its interrupt requests must occur within 100 microseconds.

The magnetic tape unit receives next priority. Although magnetic tapes transfer data much faster than the paper tape reader, they do it through the data channel; interrupts are requested only after entire blocks of information have been transferred. Interrupt requests from the paper tape reader, which occur every frame, consequently come more frequently than those of a magnetic tape unit.

The paper tape punch and Teletype keyboard (TTI) receive priority in that order, according to their speeds. Finally, Teletype output (TTO) is always masked out, since the program does not need or want to know about TTO interrupts. There is no timing problem because Teletype output can keep up with Teletype input.

Interrupts may come from two other sources in addition to the peripherals mentioned above. If the computer includes the power monitor and auto-restart feature, an interrupt will occur if power should fail. Since power fail interrupts cannot be masked out, a power failure has priority over all other peripherals. The second other source of an interrupt is a peripheral from which the program does not expect an interrupt. (For example, an interrupt from a second terminal may occur if it is used when the program is running. The program handles such a spurious interrupt merely by clearing the interrupting peripheral's Done flag and returning to the interrupted program. The Interrupt On flag is left set to 0 during this short process; consequently, spurious interrupts, like power fail interrupts, have priority equal to that of the paper tape reader, the highest-priority active peripheral.

Priority Interrupt Handler

The interrupt handler in Example Three differs from that in Example Two in a number of ways. The reason for these differences is twofold:

1. Example Three sets up and maintains a priority interrupt structure.
2. Example Three includes programming to handle spurious interrupts and to make use of the power monitor and auto-restart feature.

The interrupt handler uses the INTERRUPT ACKNOWLEDGE instruction (INTA) to identify the source of an interrupt. The device code read by the INTA instruction is used as an index into a 64-word interrupt dispatch table (IDTAB) which contains the starting addresses of the various peripheral service routines. The main interrupt handler dispatches control to the appropriate peripheral service routine by performing a "jump indirect" (JMP@) to the correct table location. Table locations which correspond to peripherals not provided for in Example Three contain the address of the spurious interrupt handler (SIH).

In order to change the current priority level when an interrupt occurs, a new interrupt priority mask must be established. In addition, the previous value of the mask must be saved so that it can be restored when the current interrupt service is finished. In Example Three, the various peripheral service routines are responsible for manipulating the old and new masks. The old mask is retrieved from the location reserved for it (MASK) and stored in the routine's save area. Then a new mask, which reflects the priority structure required by the peripheral, is loaded into AC0. The new mask is established by executing two instructions: (1) DOBS 0,CPU, which is a combination of the MASK OUT and INTERRUPT ENABLE instructions, sets up the Interrupt Disable flags in the system and turns interrupts back on, so that higher-priority peripherals may interrupt the current service routine. (2) STA 0,CMASK stores a copy of the new mask in the current mask save location. These two instructions should, in general, always be given together, so that the current value of the mask stored in memory (at CMASK) matches the states of the Interrupt Disable flags. Note also that the mask must be set up at the beginning of the program, before interrupts are enabled. In Example Three a DOBS 0,CPU instruction initializes the mask to 1 to mask out Teletype output and turns interrupts on. No interrupt can occur until after the STA 0,CMASK instruction which follows immediately.

Because a higher-priority peripheral may interrupt the service routine of a lower-priority peripheral, the first portion of the interrupt handler must be written in such a way that no information vital to the currently executing service routine is lost when a higher-priority interrupt occurs. Therefore, the information of interest must be protected before interrupts can be enabled. In any priority interrupt handler, the current mask and the return address stored in location 0 must be saved. In Example Three, it is also necessary to transfer the saved values of AC0 and AC3 from the temporary locations SAVE0 and SAVE3 to the protected area special to each peripheral service routine (TSAV0 and TSAV3 for Teletype input service, for example). (The save area for each service routine is protected because Example Three is set up so that no peripheral can interrupt itself.) It is not necessary to store the values of the accumulators or Carry in a protected area before enabling interrupts. Any higher-priority service routine that interrupts a lower-priority routine before the lower-priority routine saves the values of the accumulators it needs to use is required to ensure that, upon return to the lower-priority routine, the accumulators are restored to the values they had at the time of the higher-priority interrupt. Consequently, information in Carry or the accumulators is "safe" with respect to higher-priority interrupts.

The power fail service routine has been expanded in Example Three to make use of the auto-restart feature. When a power fail interrupt occurs, the Carry and accumulators are saved, and the computer is readied for an auto-restart. This involves placing a JUMP instruction in location 0. When auto-restart occurs, the computer automatically simulates a JMP 0 instruction (note: not a JMP @0 instruction). The restart instruction stored in location 0 (JMP @RSTAD) transfers control to the restart code (RSTRT), where the state of the machine is restored and the program is picked up where it left off when power failed. Each peripheral has a software "active" flag set when it begins an operation, so the program can check on how far the operation proceeded before power failed. Since the magnetic tape unit could have lost power anywhere within the current record it was writing, the program erases a section of tape and rewrites the entire record. Note that an auto-restart will occur only if the power switch is in the "lock" position. However, if the power switch is not in "lock", the same effect can be produced manually by starting the computer at location 0.

A routine (SIH) is included in Example Three to handle spurious interrupts. It merely clears the peripheral which caused the interrupt and then returns to the interrupted program. The peripheral is cleared by executing an NIOC instruction with the device code read by the INTA instruction given earlier.

The power fail routine, the spurious interrupt handler, and the paper tape reader service routine are given "absolute" priority while they are executing. Interrupts are not reenabled by these routines, so no other interrupt is allowed to occur. Consequently, these routines need not manipulate the priority mask nor safeguard the return address stored in location 0.

In Example Three each service routine is responsible for dismissing the interrupt on its own. This involves restoring the states of Carry, the accumulators, and the previous mask, and returning to the interrupted program. The priority mask should be changed only when interrupts are disabled. Therefore, the dismissal sequence must disable interrupts before restoring the mask, then enable them and immediately jump back to the interrupted program. In Example Three, a DOBC 0,CPU is given to change the mask and disable interrupts. Then location CMASK is restored, the last of the accumulators is restored, interrupts are enabled, and control is returned to the interrupted program.

EXAMPLE THREE

```
        .LOC   0

        0                  ;PC STORED HERE ON INTERRUPT
        IHAND              ;INTERRUPT HANDLER ADDRESS
CMASK:  0                  ;CURRENT MASK SAVE LOCATION
RSTAD:  RSTRT              ;AUTO-RESTART ADDRESS

SAVE0:  0                  ;SAVE AREA FOR MAIN INTERRUPT HANDLER AND
SAVE1:  0                  ; POWER FAIL SERVICE ROUTINE
SAVE2:  0
SAVE3:  0
SAVEC:  0                  ;PC AND CARRY SAVE LOCATION

PACTV:  0                  ;PUNCH ACTIVE FLAG
RACTV:  0                  ;READER ACTIVE FLAG
MACTV:  0                  ;MAG TAPE ACTIVE FLAG

        .LOC  20           ;AUTO-INCREMENTING LOCATIONS

        BUFF-1             ;FIRST INDEX INTO FIRST CHARACTER BUFFER
        BUFF-1             ;SECOND INDEX INTO FIRST CHARACTER BUFFER
        BUFF2-1            ;INDEX INTO SECOND CHARACTER BUFFER

        .LOC  400

;       "MAIN" PROGRAM

START:  IORST              ;CLEAR I/O DEVICES
        LDA    0,RWIND     ;REWIND MAG TAPE
        DOAS   0,MTA
        SUBZL  0,0         ;SET AC0 TO 1
        DOBS   0,CPU       ;MASK OUT TTO (BIT 15) AND TURN ON INTERRUPTS
        STA    0,CMASK     ;SAVE CURRENT MASK

CALC:   ADD    0,1         ;SIMULATE USEFUL COMPUTATION
        SUB    1,2
        INC    2,3
        NEG    3,0
        ISZ    LOC
        JMP    CALC
        JMP    CALC

LOC:    0
RWIND:  10                 ;REWIND COMMAND (SELECTS DRIVE 0)


;       INTERRUPT HANDLER

IHAND:  STA    0,SAVE0     ;SAVE AC0 AND AC3 FOR USE WITH INTA
        STA    3,SAVE3
        SKPDZ  CPU         ;CHECK FOR POWER FAIL INTERRUPT
        JMP    PFLSR
        LDA    3,AIDTB     ;GET ADDRESS OF INTERRUPT DISPATCH TABLE
        INTA   0           ;READ DEVICE CODE
        ADD    0,3         ;POINT TO CORRECT TABLE ENTRY
        JMP    @0,3        ;DISPATCH TO PERIPHERAL SERVICE ROUTINE

AIDTB:  IDTAB              ;ADDRESS OF INTERRUPT DISPATCH TABLE


;       POWER FAIL/AUTO-RESTART SERVICE ROUTINE

PFLSR:  STA    1,SAVE1     ;SAVE REMAINING ACCUMULATORS
        STA    2,SAVE2
        LDA    0,0         ;GET SAVED PC FROM LOCATION 0
```

EXAMPLE THREE (Continued)

```
        MOVL   0,0          ;APPEND CARRY AT LOW END
        STA    0,SAVEC      ;SAVE PC AND CARRY
        LDA    0,JUMP       ;SET UP RESTART INSTRUCTION IN LOCATION 0
        STA    0,0
        SKPDZ  CPU          ;LOOP HERE WAITING FOR POWER TO GO DOWN
        JMP    .-1
        JMP    RSTOR        ;IF IT COMES BACK UP GO AHEAD AND RESTORE

JUMP:   JMP    @RSTAD       ;INSTRUCTION FOR JUMPING TO RESTART CODE


RSTRT:  LDA    0,PACTV      ;WAS PTP ACTIVE?
        MOV#   0,0,SZR
        NIOS   PTP          ;IF YES, RESTART IT
        LDA    0,RACTV      ;WAS PTR ACTIVE?
        MOV#   0,0,SZR
        NIOS   PTR          ;IF YES, RESTART IT
        LDA    0,MACTV      ;WAS MTA ACTIVE?
        MOV#   0,0,SZR
        JSR    @AMTAR       ;IF YES, GO RESTART IT

RSTOR:  LDA    0,CMASK      ;RESTORE CURRENT MASK
        DOBC   0,CPU        ;MSKO AND INTDS
        LDA    0,SAVEC      ;RESTORE CARRY AND SAVED PC
        MOVZR  0,0
        STA    0,0          ;SET UP RETURN ADDRESS
        LDA    0,SAVE0      ;RESTORE ACCUMULATORS
        LDA    1,SAVE1
        LDA    2,SAVE2
        LDA    3,SAVE3
        INTEN               ;ENABLE INTERRUPTS
        JMP    @0           ;RESTART PROGRAM

AMTAR:  MTAAR               ;ADDRESS OF ROUTINE TO RESTART MAG TAPE


;       ROUTINE TO HANDLE SPURIOUS INTERRUPTS

SIH:    LDA    3,NIOC0      ;CLEAR DEVICE THAT CAUSED THE INTERRUPT:
        ADD    0,3          ;FORM AN NIOC INSTRUCTION WITH APPROPRIATE DEVICE CODE
        STA    3,.+1        ;STORE THE INSTRUCTION
        0                   ;EXECUTE THE INSTRUCTION
        LDA    0,SAVE0      ;RESTORE ACCUMULATORS
        LDA    3,SAVE3
        INTEN               ;ENABLE INTERRUPTS
        JMP    @0           ;RETURN TO INTERRUPTED PROGRAM

NIOC0:  NIOC   0            ;USED TO CREATE AN NIOC <DEVICE> INSTRUCTION


;       TELETYPE INPUT SERVICE ROUTINE

TTISR:  LDA    0,SAVE0      ;SAVE AC0 AND AC3 IN TTI SAVE AREA
        STA    0,TSAV0
        LDA    0,SAVE3
        STA    0,TSAV3
        LDA    0,CMASK      ;SAVE CURRENT MASK
        STA    0,TSAVM
        LDA    3,0          ;LOAD RETURN ADDRESS INTO AC3
        LDA    0,TMASK      ;GET NEW MASK
        DOBS   0,CPU        ;MSKO AND INTEN
        STA    0,CMASK      ;UPDATE CURRENT MASK SAVE LOCATION
        MOVL   3,3          ;APPEND CARRY TO LOW END OF RETURN ADDRESS
        STA    3,TSAVC      ;SAVE RETURN ADDRESS AND CARRY

READC:  DIAC   0,TTI        ;READ THE CHARACTER AND CLEAR DONE
        LDA    3,C177       ;DROP PARITY BIT
        AND    3,0
```

EXAMPLE THREE (Continued)

```
STORE:  STA   0,@20      ;STORE CHARACTER IN BUFF
        LDA   3,CNTLR    ;CHECK FOR CONTROL R
        SUB#  3,0,SNR
        LDA   0,BELL     ;ECHO CONTROL R BY RINGING BELL
        SKPBZ TTO        ;TTO READY?
        JMP   .-1        ;NO, TRY AGAIN
        DOAS  0,TTO      ;YES, ECHO CHARACTER
        LDA   3,CR       ;IS CHARACTER A CARRIAGE RETURN?
        SUB#  3,0,SNR
        JMP   CRET       ;YES, GO PROCESS IT
        DSZ   CHCNT      ;NO, DECREMENT CHCNT TO CHECK FOR LINE OVERFLOW
        JMP   TTIDS      ;NO OVERFLOW, GO DISMISS INTERRUPT
        MOV   3,0        ;LINE IS FULL, LOAD A CR AND
        JMP   STORE      ; JUMP BACK TO INSERT IT INTO BUFF
CRET:   LDA   0,LF       ;ADD LINE FEED TO BUFF
        STA   0,@20
        SKPBZ TTO        ;WAIT UNTIL TTO READY
        JMP   .-1
        DOAS  0,TTO      ;GIVE LINE FEED
        LDA   0,PCCNT    ;WAIT IF PUNCHING OF PREVIOUS LINE IS NOT DONE
        MOV#  0,0,SZR
        JMP   .-2
        LDA   0,20       ;CALCULATE NUMBER OF CHARACTERS IN BUFF
        LDA   3,ABUFF
        SUB   3,0
        STA   0,PCCNT    ;INITIALIZE PUNCH CHARACTER COUNTER
        LDA   0,MAX      ;REINITIALIZE CHARACTER BUFFER:
        STA   0,CHCNT    ; INITIALIZE CHARACTER COUNTER
        STA   3,20       ; INITIALIZE CHARACTER BUFFER POINTERS IN
        STA   3,21       ; AUTO-INCREMENTING LOCATIONS 20 AND 21
        SUB   0,0        ;FIRST PUT OUT 8 FRAMES OF BLANK LEADER
        INTDS            ;MAKE SURE THAT PUNCH STARTED AND FLAG SET TOGETHER
        DOAS  0,PTP      ;OUTPUT NULL CHARACTER
        ISZ   PACTV      ;SET PUNCH ACTIVE FLAG
        INTEN
        STA   0,CCHAR    ;SAVE NULL AS CURRENT CHARACTER

TTIDS:  LDA   0,TSAVC    ;TTI INTERRUPT DISMISSAL
        MOVZR 0,0        ;RESTORE CARRY AND SET UP RETURN ADDRESS
        STA   0,TSAVC
        LDA   3,TSAV3    ;RESTORE AC3
        LDA   0,TSAVM    ;GET PREVIOUS MASK
        DOBC  0,CPU      ;MSKO AND INTDS
        STA   0,CMASK    ;RESTORE MASK SAVE LOCATION
        LDA   0,TSAV0    ;RESTORE AC0
        INTEN            ;ENABLE INTERRUPTS
        JMP   @TSAVC     ;RETURN TO INTERRUPTED PROGRAM

TSAV0:  0                ;TTI SAVE AREA: AC0
TSAV3:  0                ;AC3
TSAVC:  0                ;RETURN ADDRESS AND CARRY
TSAVM:  0                ;CURRENT MASK

TMASK:  3                ;MASKS OUT TTI AND TTO

C177:   177              ;MASK FOR DELETING PARITY BIT
CR:     15               ;CARRIAGE RETURN
LF:     12               ;LINE FEED
CNTLR:  22               ;CONTROL R
BELL:   7                ;TELETYPE BELL


ABUFF:  BUFF-1           ;ADDRESS OF CHARACTER BUFFER - 1
MAX:    110              ;MAXIMUM OF 72 CHARACTERS INPUT PER LINE
CHCNT:  110              ;CHARACTER COUNTER (COUNTS DOWN FROM MAX)


PCCNT:  0                ;PUNCH CHARACTER COUNTER
PRCNT:  10               ;PUNCH REPEAT COUNTER
NRPT:   10               ;NUMBER OF TIMES TO PUNCH EACH CHARACTER
CCHAR:  0                ;CURRENT CHARACTER SAVE LOCATION
```

EXAMPLE THREE (Continued)

```
;       PAPER TAPE PUNCH SERVICE ROUTINE

PTPSR:  LDA     0,SAVE0     ;SAVE AC0 AND AC3 IN PTP SAVE AREA
        STA     0,PSAV0
        LDA     0,SAVE3
        STA     0,PSAV3
        LDA     0,CMASK     ;SAVE CURRENT MASK
        STA     0,PSAVM
        SUB     0,0         ;CLEAR PUNCH ACTIVE FLAG
        STA     0,PACTV
        LDA     3,0         ;LOAD RETURN ADDRESS INTO AC3
        LDA     0,PMASK     ;GET NEW MASK
        DOBS    0,CPU       ;MSKO AND INTEN
        STA     0,CMASK     ;UPDATE CURRENT MASK SAVE
        MOVL    3,3         ;APPEND CARRY TO LOW END OF RETURN ADDRESS
        STA     3,PSAVC     ;SAVE RETURN ADDRESS AND CARRY

GCHAR:  LDA     0,CCHAR     ;GET CURRENT CHARACTER
        LDA     3,CNTLR     ;IS IT CONTROL R?
        SUB     0,3,SNR     ;(AC3 BECOMES 0 IF CHARACTER IS CONTROL R)
        JMP     PTRDR       ;YES, INPUT FROM READER
        DSZ     PRCNT       ;NO, DECREMENT PUNCH REPEAT COUNTER
        JMP     SAMEC       ;GO PUNCH SAME CHARACTER AGAIN
CONTP:  LDA     0,NRPT      ;REINITIALIZE PUNCH REPEAT COUNTER
        STA     0,PRCNT
        DSZ     PCCNT       ;DECREMENT PUNCH CHARACTER COUNTER
        JMP     NEWC        ;GO PUNCH NEW CHARACTER
        NIOC    PTP         ;ALL DONE, CLEAR PUNCH
        JMP     PTPDS       ;GO DISMISS INTERRUPT

NEWC:   LDA     0,@21       ;GET NEW CHARACTER
        STA     0,CCHAR     ;UPDATE CURRENT CHARACTER SAVE
SAMEC:  INTDS               ;MAKE SURE PUNCH STARTED AND FLAG SET TOGETHER
        DOAS    0,PTP       ;OUTPUT CHARACTER TO PUNCH
        ISZ     PACTV       ;SET PUNCH ACTIVE FLAG
        INTEN
        JMP     PTPDS       ;GO DISMISS INTERRUPT

PTRDR:  ISZ     RBCNT       ;INCREMENT READER BLOCK COUNTER
        LDA     0,RBCNT     ;IS THIS THE FIRST PTR REQUEST?
        ADC#    3,0,SZR     ;(SKIP IF AC0 = 1)
        JMP     CONTP       ;NO, CONTINUE PUNCHING WITH NEXT CHARACTER
        INTDS               ;MAKE SURE READER STARTED AND FLAG SET TOGETHER
        NIOS    PTR         ;START READER
        ISZ     RACTV       ;SET READER ACTIVE FLAG
        INTEN
        LDA     0,ABUF2     ;INITIALIZE SECOND CHARACTER BUFFER
        STA     0,22        ;USE AUTO-INCREMENT LOCATION 22 AS INDEX
        JMP     CONTP       ;CONTINUE PUNCHING WITH NEXT CHARACTER

PTPDS:  LDA     0,PSAVC     ;PTP INTERRUPT DISMISSAL
        MOVZR 0,0           ;RESTORE CARRY AND SET UP RETURN ADDRESS
        STA     0,PSAVC
        LDA     3,PSAV3     ;RESTORE AC3
        LDA     0,PSAVM     ;GET PREVIOUS MASK
        DOBC    0,CPU       ;MSKO AND INTDS
        STA     0,CMASK     ;RESTORE PREVIOUS MASK SAVE
        LDA     0,PSAV0     ;RESTORE AC0
        INTEN               ;ENABLE INTERRUPTS
        JMP     @PSAVC      ;RETURN TO INTERRUPTED PROGRAM


PSAV0:  0                   ;PTP SAVE AREA: AC0
PSAV3:  0                   ;AC3
PSAVC:  0                   ;RETURN ADDRESS AND CARRY
PSAVM:  0                   ;CURRENT MASK

PMASK:  7                   ;MASKS OUT PTP, TTI, AND TTO

RBCNT:  0                   ;NUMBER OF BLOCKS LEFT TO READ FROM PTR
```

EXAMPLE THREE (Continued)

```
ABUF2:  BUFF2-1         ;ADDRESS OF SECOND CHARACTER BUFFER - 1


;       PAPER TAPE READER SERVICE ROUTINE

PTRSR:  STA    1,RSAV1    ;SAVE ONE MORE ACCUMULATOR AND CARRY
        MOVL   0,0        ;SINCE PTR HAS HIGHEST PRIORITY, LEAVE INTERRUPTS
        STA    0,RSAVC    ; OFF, AND DON'T BOTHER WITH RETURN ADDRESS OR MASK

        DIAC   0,PTR      ;READ FRAME AND CLEAR DONE
        LDA    3,CHECK    ;GET PREVIOUS FRAME READ
        STA    0,CHECK    ;SAVE NEW FRAME
        MOV#   0,0,SNR    ;IS NEW FRAME = 0?
        JMP    ZEROF      ;YES
        STA    0,@22      ;NO, SAVE IT IN BUFF2
        ISZ    RFCNT      ;INCREMENT READER FRAME COUNTER (NEVER SKIP)
SPTR:   NIOS   PTR        ;START READER FOR NEXT FRAME
        JMP    PTRDS      ;GO DISMISS INTERRUPT

ZEROF:  MOV#   3,3,SNR    ;FRAME IS ZERO, WAS PREVIOUS ONE ALSO ZERO?
        JMP    SPTR       ;YES, IGNORE LEADING BLANK FRAMES
        STA    0,@22      ;NO, DONE WITH THIS BLOCK, MARK WITH 0 FRAME
        ISZ    RFCNT      ;INCREMENT READER FRAME COUNTER
        SUB    0,0        ;CLEAR READER ACTIVE FLAG
        STA    0,RACTV
        LDA    1,BBUF2    ;AC1 POINTS TO BEGINNING OF BUFF2
        JSR    MWRIT      ;GO WRITE A BLOCK ON MAG TAPE

PTRDS:  LDA    0,RSAVC    ;RESTORE CARRY AND ACCUMULATORS
        MOVR   0,0
        LDA    1,RSAV1
        LDA    0,SAVE0
        LDA    3,SAVE3
        INTEN             ;ENABLE INTERRUPTS
        JMP    @0         ;RETURN TO INTERRUPTED PROGRAM

RSAV1:  0                 ;READER SAVE AREA: AC1
RSAVC:  0                 ;CARRY SAVE LOCATION

BBUF2:  BUFF2             ;ADDRESS OF BUFF2
RFCNT:  0                 ;READER FRAME COUNTER
CHECK:  0                 ;LAST FRAME INPUT FROM READER, USED IN IGNORING
                          ; LEADING BLANK FRAMES


;       MAG TAPE SERVICE ROUTINE

MTASR:  LDA    0,SAVE0    ;SAVE AC0 AND AC3 IN MTA SAVE AREA
        STA    0,MSAV0
        LDA    0,SAVE3
        STA    0,MSAV3
        LDA    0,CMASK    ;SAVE CURRENT MASK
        STA    0,MSAVM
        SUB    0,0        ;CLEAR MAG TAPE ACTIVE FLAG
        STA    0,MACTV
        LDA    3,0        ;LOAD RETURN ADDRESS INTO AC3
        LDA    0,MMASK    ;GET NEW MASK
        DOBS   0,CPU      ;MSKO AND INTEN
        STA    0,CMASK    ;UPDATE CURRENT MASK SAVE
        MOVL   3,3        ;APPEND CARRY TO LOW END OF RETURN ADDRESS
        STA    3,MSAVC    ;SAVE RETURN ADDRESS AND CARRY
        STA    1,MSAV1    ;SAVE AC1

        DIA    0,MTA      ;READ MAG TAPE STATUS
        MOVL#  0,0,SZC    ;CHECK FOR ERRORS
        JMP    ERROR
        LDA    0,RFCNT    ;FETCH READER FRAME COUNTER
```

EXAMPLE THREE (Continued)

```
         MOV#   0,0,SZR    ;MORE WRITING TO DO?
         JMP    MOREM      ;YES
         NIOC   MTA        ;NO, CLEAR MAG TAPE
         DSZ    RBCNT      ;DECREMENT AND CHECK READER BLOCK COUNTER
         JMP    MORER      ;MORE READING TO DO

MTADS:   LDA    0,MSAVC    ;DISMISS MTA INTERRUPT
         MOVZR  0,0        ;RESTORE CARRY AND SET UP RETURN ADDRESS
         STA    0,MSAVC
         LDA    1,MSAV1    ;RESTORE AC1
         LDA    3,MSAV3    ;RESTORE AC3
         LDA    0,MSAVM    ;GET PREVIOUS MASK
         DOBC   0,CPU      ;MSKO AND INTDS
         STA    0,CMASK    ;RESTORE MASK SAVE LOCATION
         LDA    0,MSAV0    ;RESTORE AC0
         INTEN             ;ENABLE INTERRUPTS
         JMP    @MSAVC     ;RETURN TO INTERRUPTED PROGRAM

MOREM:   DIB    1,MTA      ;READ ADDRESS OF NEXT WORD TO WRITE FROM BUFF2
         JSR    MWRIT      ;GO WRITE NEXT BLOCK ON MAG TAPE
         JMP    MTADS      ;GO DISMISS INTERRUPT

MORER:   INTDS             ;MAKE SURE READER STARTED AND FLAG SET TOGETHER
         NIOS   PTR        ;START READER
         ISZ    RACTV      ;SET READER ACTIVE FLAG
         INTEN
         LDA    0,ABUF2    ;REINITIALIZE BUFF2
         STA    0,22
         JMP    MTADS      ;GO DISMISS INTERRUPT

ERROR:   HALT              ;ERROR PROCESSING WOULD NORMALLY GO HERE

MSAV0:   0                 ;MTA SAVE AREA: AC0
MSAV1:   0                 ;AC1
MSAV3:   0                 ;AC3
MSAVC:   0                 ;RETURN ADDRESS AND CARRY
MSAVM:   0                 ;CURRENT MASK

MMASK:   47                ;MASKS OUT MTA, PTP, TTI, AND TTO


;        MWRIT WRITES A BLOCK OF DATA IN MEMORY ON MAG TAPE DRIVE 0
;           SIZE OF BLOCK IS -BLKSZ
;           ADDRESS OF FIRST WORD OF BLOCK IS CONTAINED IN AC1 ON ENTRY

MWRIT:   DOB    1,MTA      ;LOAD MEMORY ADDRESS COUNTER
         STA    1,LSTAD    ;SAVE LAST ADDRESS USED IN WRITING TO MAG TAPE
         LDA    1,BLKSZ    ;FETCH NEGATIVE BLOCK SIZE
         DOC    1,MTA      ;LOAD WORD COUNTER
         LDA    0,WRCOM    ;FETCH WRITE COMMAND
         INTDS             ;MAKE SURE MAG TAPE STARTED AND FLAG SET TOGETHER
         DOAS   0,MTA      ;INITIATE WRITE OPERATION
         ISZ    MACTV      ;SET MAG TAPE ACTIVE FLAG
         INTEN
         LDA    0,RFCNT    ;NOW UPDATE READER FRAME COUNTER
         ADDZ   1,0,SNC    ;DECREASE IT BY BLOCK SIZE
         SUB    0,0        ;IF RFCNT WAS < BLKSZ, SET RFCNT TO 0
         STA    0,RFCNT    ;SAVE NEW VALUE
         JMP    0,3        ;RETURN TO CALLER

BLKSZ:   -144              ;(NEGATIVE) BLOCK SIZE OF 100 DECIMAL
WRCOM:   50                ;WRITE COMMAND (SELECTS DRIVE 0)
LSTAD:   0                 ;LAST ADDRESS USED IN WRITING TO MAG TAPE
ERCOM:   70                ;ERASE COMMAND (SELECTS DRIVE 0)
```

EXAMPLE THREE (Continued)

```
;       MTAAR RESTARTS MAG TAPE AFTER POWER FAIL
;          SEVERAL INCHES OF TAPE ARE ERASED, THEN THE RECORD
;          WHICH WAS BEING WRITTEN WHEN POWER FAILED IS REWRITTEN

MTAAR:  LDA    0,ERCOM    ;ERASE SECTION OF TAPE
        DOAS   0,MTA
        SKPDN  MTA        ;WAIT TILL MTA DONE
        JMP    .-1
        LDA    0,RECNT    ;SET RECNT BACK TO ITS PRIOR VALUE
        LDA    1,BLKSZ
        SUB    1,0
        STA    1,RECNT
        LDA    1,LSTAD    ;FETCH ADDRESS OF RECORD TO BE REWRITTEN
        JMP    MWRIT      ;GO WRITE RECORD (MWRIT RETURNS DIRECTLY TO
                          ;AUTO-RESTART CODE)



;       CHARACTER BUFFERS

BUFF:   .BLK   112        ;ALLOW FOR 72 CHARACTERS + CR + LF
BUFF2:  .BLK   1120       ;ALLOW FOR UP TO 74*8=592 FRAMES


;       INTERRUPT DISPATCH TABLE

IDTAB:  SIH               ;DEVICE CODE 0
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        TTISR             ;DEVICE CODE 10 = TTI
        SIH
        PTRSR             ;DEVICE CODE 12 = PTR
        PTPSR             ;DEVICE CODE 13 = PTP
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        MTASR             ;DEVICE CODE 22 = MTA
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
```

EXAMPLE THREE (Continued)

```
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH
        SIH             ;DEVICE CODE 77

        .END  START
```

EXAMPLE FOUR

Example Four performs the same functions as Example Three. The majority of the interrupt handling functions are accomplished using the VECTOR instruction (VCT) and the stack manipulation instructions of the ECLIPSE computer. Stack context is changed from the (possible) user stack to a "systems" stack when a base level interrupt occurs. Other nested interrupts do not require a stack context change as the active stack is the systems stack.

Interrupt dismissal involves returning to a lower level service routine using the POP BLOCK instruction (POPB) after checking to determine the return is not to base level. If the return is to base level, the RESTORE instruction (RSTR) also effects a stack context change to reactivate the user stack.

EXAMPLE FOUR FOR USE WITH THE ECLIPSE COMPUTER

```
         .LOC    0

;        RESERVED STORAGE LOCATIONS - CANNOT BE RELOCATED

         0                  ;PC STORED HERE ON INTERRUPT
IHA:     IHAND              ;INTERRUPT HANDLER ADDRESS
SCHA:    SYSER              ;SYSTEM CALL HANDLER ADDRESS (NOT USED HERE)
PFHA:    SYSER              ;PROTECTION FAULT HANDLER ADDRESS (NOT USED HERE)
VSP:     VSTK-1             ;ADDRESS OF TOP OF VECTOR STACK
CMASK:   0                  ;CURRENT MASK SAVE LOCATION
VSL:     VSTK-1+VSLEN       ;VECTOR STACK LIMIT
VSFHA:   SYSER              ;VECTOR STACK FAULT HANDLER ADDRESS

;        ARBITRARY PAGE ZERO STORAGE LOCATIONS FOR THIS EXAMPLE

RSTAD:   RSTRT              ;AUTO-RESTART ADDRESS
SYSER:   HALTA 0            ;THIS EXAMPLE DOES NOT HANDLE SYSTEM CALLS,
                            ; PROTECTION FAULTS, OR VECTOR STACK FAULTS
PACTV:   0                  ;PUNCH ACTIVE FLAG
RACTV:   0                  ;READER ACTIVE FLAG
MACTV:   0                  ;MAG TAPE ACTIVE FLAG


         .LOC   20          ;AUTO-INCREMENTING LOCATIONS

         BUFF-1             ;FIRST INDEX INTO FIRST CHARACTER BUFFER
         BUFF-1             ;SECOND INDEX INTO FIRST CHARACTER BUFFER
         BUFF2-1            ;INDEX INTO SECOND CHARACTER BUFFER


         .LOC   401

;        VECTOR STACK
;
;        DEFINE THE VECTOR STACK LENGTH
         VSLEN = 5*6+4      ;ALLOWS FOR 6-WORD BLOCKS FOR UP TO 5 INTERRUPT
                            ; LEVELS (TTI, PTP, MTA, PTR, PFL) PLUS ROOM TO
                            ; SAVE USER STACK PARAMETERS (LOCATIONS 40-43)

VSTK:    .BLK   VSLEN+13    ;RESERVE STACK AREA (ALLOW 11 EXTRA WORDS IN
                            ; CASE OF STACK OVERFLOW)

;        "MAIN" PROGRAM

START:   IORST              ;CLEAR I/O DEVICES
         LDA    0,RWIND     ;REWIND MAG TAPE
         DOAS   0,MTA
         SUBZL  0,0         ;SET AC0 TO 1
         DOBS   0,CPU       ;MASK OUT TTO (BIT 15) AND TURN ON INTERRUPTS
         STA    0,CMASK     ;SAVE CURRENT MASK
```

EXAMPLE FOUR (Continued)

```
CALC:   ADD    0,1           ;SIMULATE USEFUL COMPUTATION
        SUB    1,2           ; (MAIN PROGRAM MAY USE A STACK OF ITS OWN)
        INC    2,3
        NEG    3,0
        ISZ    LOC
        JMP    CALC
        JMP    CALC

LOC:    0
RWIND:  10                   ;REWIND COMMAND (SELECTS DRIVE 0)



;       INTERRUPT HANDLER

IHAND:  ISZ    LEVEL         ;HAS BASE LEVEL JUST BEEN INTERRUPTED?
        JMP    NSCH          ;NO
        VCT    @VTAB         ;YES, VECTOR WITH POSSIBLE STACK CHANGE
NSCH:   VCT    VTAB          ;VECTOR WITH NO STACK CHANGE


;       COMMON INTERRUPT DISMISSAL

DISMS:  LDA    1,LEVEL       ;GET CURRENT INTERRUPT LEVEL
        SBI    1,1           ;DECREASE IT BY 1
        POP    0,0           ;GET OLD MASK
        DOBC   0,CPU         ;RESTORE OLD MASK AND DISABLE INTERRUPTS
        STA    0,CMASK       ;RESTORE MASK SAVE LOCATION
        STA    1,LEVEL       ;SET NEW INTERRUPT LEVEL
        COM#   1,1,SNR       ;GOING BACK TO BASE LEVEL?
        JMP    .+3           ;YES
        INTEN                ;NO, LEAVE VECTOR STACK AS IS, WILL BE
        POPB                 ; RETURNING TO SOME OTHER SERVICE ROUTINE
        INTEN                ;RESTORE MAIN PROGRAM STACK ON WAY BACK TO
        RSTR                 ; BASE LEVEL (I.E., MAIN PROGRAM)

LEVEL:  -1                   ;INTERRUPT LEVEL COUNTER



;       POWER FAIL/AUTO-RESTART SERVICE ROUTINE

PFLCT:  @PFLSR               ;ADDRESS OF POWER FAIL SERVICE ROUTINE (PUSH BIT = 1)
        0                    ;(DON'T BOTHER CHANGING MASK, SINCE INTERRUPTS
                             ; WILL BE TURNED OFF IMMEDIATELY AT PFLSR)

PFLSR:  INTDS                ;DISABLE INTERRUPTS
        SKPDZ  CPU           ;IS THIS REALLY A POWER FAIL?
        JMP    REAL          ;YES
        ISZ    PFCNT         ;NO, COUNT THIS DECEPTION
        JMP    DISMS         ;GO DISMISS INTERRUPT
        HALTA  0             ;HALT IF IT HAPPENS 65,536 TIMES

REAL:   LDA    0,JUMP        ;SET UP RESTART INSTRUCTION IN LOCATION 0
        STA    0,0
        SKPDZ  CPU           ;LOOP HERE WAITING FOR POWER TO GO DOWN
        JMP    .-1
        JMP    RSTOR         ;IF IT COMES BACK UP, GO AHEAD AND RESTORE

RSTRT:  LDA    0,PACTV       ;WAS PTP ACTIVE?
        MOV#   0,0,SZR
        NIOS   PTP           ;IF YES, RESTART IT
        LDA    0,RACTV       ;WAS PTR ACTIVE?
        MOV#   0,0,SZR
        NIOS   PTR           ;IF YES, RESTART IT
        LDA    0,MACTV       ;WAS MTA ACTIVE?
        MOV#   0,0,SZR
        JSR    @AMTAR        ;IF YES, GO RESTART IT
```

EXAMPLE FOUR (Continued)

```
RSTOR:  JMP    DISMS       ;GO RESTORE AND RESTART PROGRAM


PFCNT:  0                  ;COUNTS NUMBER OF SPURIOUS INTERRUPTS FROM DEVICE 0
JUMP:   JMP    @RSTAD      ;INSTRUCTION FOR JUMPING TO RESTART CODE
AMTAR:  MTAAR              ;ADDRESS OF ROUTINE TO RESTART MAG TAPE



;       ROUTINE TO HANDLE SPURIOUS INTERRUPTS

SIH:    DSZ    LEVEL       ;DECREMENT INTERRUPT LEVEL COUNTER
        JMP    .+1
        STA    0,SAVE0     ;SAVE THREE ACCUMULATORS
        STA    2,SAVE2
        STA    3,SAVE3
        INTA   0           ;READ A DEVICE CODE
        LDA    2,NSCH+1    ;GET ADDRESS OF VECTOR TABLE
        ADD    0,2         ;INDEX INTO VECTOR TABLE
        LDA    2,0,2       ;GET TABLE ENTRY
        LDA    3,ASIH      ;IS THIS A VALID DEVICE?
        SUB#   2,3,SZR
        JMP    NEWDV       ;YES, FORGET ABOUT SPURIOUS INTERRUPT, GO DO NEW ONE
        LDA    2,NIOCO     ;THIS DEVICE UNDEFINED, TRY TO CLEAR IT
        ADD    0,2         ;FORM PROPER NIOC INSTRUCTION
        LDA    3,SKIP      ;FORM PROPER SKPDN INSTRUCTION
        ADD    0,3
        LDA    0,TRIES     ;INITIALIZE COUNTER FOR NUMBER OF TRIES
TRY:    XCT    2           ;EXECUTE NIOC INSTRUCTION
        XCT    3           ;EXECUTE SKPDN INSTRUCTION
        JMP    CLEAR       ;DEVICE IS CLEAR, GO DISMISS
        INC    0,0,SZR     ;COUNT THIS TRY
        JMP    TRY         ;GO TRY AGAIN
        HALTA  2           ;HALT SHOWING FUTILE NIOC INSTRUCTION

NEWDV:  LDA    0,SAVE0     ;RESTORE ACCUMULATORS
        LDA    2,SAVE2
        LDA    3,SAVE3
        JMP    IHAND       ;GO TO INTERRUPT HANDLER

CLEAR:  LDA    0,SAVE0     ;RESTORE ACCUMULATORS
        LDA    2,SAVE2
        LDA    3,SAVE3
        INTEN              ;ENABLE INTERRUPTS
        JMP    @0          ;RETURN TO INTERRUPTED PROGRAM


SAVE0:  0                  ;SAVE AREA FOR SPURIOUS INTERRUPT HANDLER
SAVE2:  2
SAVE3:  3

ASIH:   SIH                ;ADDRESS OF SPURIOUS INTERRUPT HANDLER
NIOCO:  NIOC   0           ;SKELETAL NIOC INSTRUCTION
SKIP:   SKPDN  0           ;SKELETAL SKPDN INSTRUCTION
TRIES:  -4000              ;TRY CLEARING DEVICE 2048 TIMES BEFORE GIVING UP



;       TELETYPE INPUT SERVICE ROUTINE

TTICT:  @TTISR             ;TTI DEVICE CONTROL TABLE (PUSH BIT = 1)
        3                  ;MASKS OUT TTI, TTO

TTISR:
READC:  DIAC   0,TTI       ;READ THE CHARACTER AND CLEAR DONE
        LDA    3,C177      ;DROP PARITY BIT
        AND    3,0
STORE:  STA    0,@20       ;STORE CHARACTER IN BUFF
        LDA    3,CNTLR     ;CHECK FOR CONTROL R
        SUB#   3,0,SNR
        LDA    0,BELL      ;ECHO CONTROL R BY RINGING BELL
```

EXAMPLE FOUR (Continued)

```
        SKPBZ  TTO             ;TTO READY?
        JMP    .-1             ;NO, TRY AGAIN
        DOAS   0,TTO           ;YES, ECHO CHARACTER
        LDA    3,CR            ;IS CHARACTER A CARRIAGE RETURN?
        SUB#   3,0,SNR
        JMP    CRET            ;YES, GO PROCESS IT
        DSZ    CHCNT           ;NO, DECREMENT CHCNT TO CHECK FOR LINE OVERFLOW
        JMP    TTIDS           ;NO OVERFLOW, GO DISMISS INTERRUPT
        MOV    3,0             ;LINE IS FULL, LOAD A CR AND
        JMP    STORE           ; JUMP BACK TO INSERT IT INTO BUFF
CRET:   LDA    0,LF            ;ADD LINE FEED TO BUFF
        STA    0,@20
        SKPBZ  TTO             ;WAIT UNTIL TTO READY
        JMP    .-1
        DOAS   0,TTO           ;GIVE LINE FEED
        LDA    0,PCCNT         ;WAIT IF PUNCHING OF PREVIOUS LINE IS NOT DONE
        MOV#   0,0,SZR
        JMP    .-2
        LDA    0,20            ;CALCULATE NUMBER OF CHARACTERS IN BUFF
        LDA    3,ABUFF
        SUB    3,0
        STA    0,PCCNT         ;INITIALIZE PUNCH CHARACTER COUNTER
        LDA    0,MAX           ;REINITIALIZE CHARACTER BUFFER:
        STA    0,CHCNT         ; INITIALIZE CHARACTER COUNTER
        STA    3,20            ; INITIALIZE CHARACTER BUFFER POINTERS IN
        STA    3,21            ; AUTO-INCREMENTING LOCATIONS 20 AND 21
        SUB    0,0             ;FIRST PUT OUT 8 FRAMES OF BLANK LEADER
        INTDS                  ;MAKE SURE THAT PUNCH STARTED AND FLAG SET TOGETHER
        DOAS   0,PTP           ;OUTPUT NULL CHARACTER
        ISZ    PACTV           ;SET PUNCH ACTIVE FLAG
        INTEN
        STA    0,CCHAR         ;SAVE NULL AS CURRENT CHARACTER

TTIDS:  JMP    @.+1            ;GO DISMISS INTERRUPT
        DISMS


C177:   177                    ;MASK FOR DELETING PARITY BIT
CR:     15                     ;CARRIAGE RETURN
LF:     12                     ;LINE FEED
CNTLR:  22                     ;CONTROL R
BELL:   7                      ;TELETYPE BELL


ABUFF:  BUFF-1                 ;ADDRESS OF CHARACTER BUFFER - 1
MAX:    110                    ;MAXIMUM OF 72 CHARACTERS INPUT PER LINE
CHCNT:  110                    ;CHARACTER COUNTER (COUNTS DOWN FROM MAX)
PCCNT:  0                      ;PUNCH CHARACTER COUNTER
PRCNT:  10                     ;PUNCH REPEAT COUNTER
NRPT:   10                     ;NUMBER OF TIMES TO PUNCH EACH CHARACTER
CCHAR:  0                      ;CURRENT CHARACTER SAVE LOCATION
```

EXAMPLE FOUR (Continued)

```
;       PAPER TAPE PUNCH SERVICE ROUTINE

PTPCT:  @PTPSR                  ;PTP DEVICE CONTROL TABLE (PUSH BIT = 1)
        7                       ;MASKS OUT PTP, TTI, AND TTO

PTPSR:  INTDS                   ;DISABLE INTERRUPTS BEFORE IT'S TOO LATE
        SUB     0,0             ;CLEAR PUNCH ACTIVE FLAG
        STA     0,PACTV
        INTEN                   ;ENABLE INTERRUPTS FOR REAL


GCHAR:  LDA     0,CCHAR       ;GET CURRENT CHARACTER
        LDA     3,CNTLR       ;IS IT CONTROL R?
        SUB     0,3,SNR       ;(AC3 BECOMES 0 IF CHARACTER IS CONTROL R)
        JMP     PTRDR         ;YES, INPUT FROM READER
        DSZ     PRCNT         ;NO, DECREMENT PUNCH REPEAT COUNTER
        JMP     SAMEC         ;GO PUNCH SAME CHARACTER AGAIN
CONTP:  LDA     0,NRPT        ;REINITIALIZE PUNCH REPEAT COUNTER
        STA     0,PRCNT
        DSZ     PCCNT         ;DECREMENT PUNCH CHARACTER COUNTER
        JMP     NEWC          ;GO PUNCH NEW CHARACTER
        NIOC    PTP           ;ALL DONE, CLEAR PUNCH
        JMP     PTPDS         ;GO DISMISS INTERRUPT

NEWC:   LDA     0,@21         ;GET NEW CHARACTER
        STA     0,CCHAR       ;UPDATE CURRENT CHARACTER SAVE
SAMEC:  INTDS                 ;MAKE SURE PUNCH STARTED AND FLAG SET TOGETHER
        DOAS    0,PTP         ;OUTPUT CHARACTER TO PUNCH
        ISZ     PACTV         ;SET PUNCH ACTIVE FLAG
        INTEN
        JMP     PTPDS         ;GO DISMISS INTERRUPT

PTRDR:  ISZ     RBCNT         ;INCREMENT READER BLOCK COUNTER
        LDA     0,RBCNT       ;IS THIS THE FIRST PTR REQUEST?
        ADC#    3,0,SZR       ;(SKIP IF AC0 = 1)
        JMP     CONTP         ;NO, CONTINUE PUNCHING WITH NEXT CHARACTER
        INTDS                 ;MAKE SURE READER STARTED AND FLAG SET TOGETHER
        NIOS    PTR           ;START READER
        ISZ     RACTV         ;SET READER ACTIVE FLAG
        INTEN
        LDA     0,ABUF2       ;INITIALIZE SECOND CHARACTER BUFFER
        STA     0,22          ;USE AUTO-INCREMENT LOCATION 22 AS INDEX
        JMP     CONTP         ;CONTINUE PUNCHING WITH NEXT CHARACTER

PTPDS:  JMP     @.+1          ;GO DISMISS INTERRUPT
        DISMS


RBCNT:  0                       ;NUMBER OF BLOCKS LEFT TO READ FROM PTR
ABUF2:  BUFF2-1                 ;ADDRESS OF SECOND CHARACTER BUFFER - 1


;       PAPER TAPE READER SERVICE ROUTINE

PTRCT:  @PTRSR                  ;PTR DEVICE CONTROL TABLE (PUSH BIT = 1)
        63                      ;MASKS OUT PTR, MTA, TTI, TTO

PTRSR:  INTDS                   ;DISABLE INTERRUPTS
        SUB     0,0             ;CLEAR READER ACTIVE FLAG
        STA     0,RACTV
        INTEN                   ;ENABLE INTERRUPTS
        DIAC    0,PTR           ;READ FRAME AND CLEAR DONE
        LDA     3,CHECK         ;GET PREVIOUS FRAME READ
        STA     0,CHECK         ;SAVE NEW FRAME
        MOV#    0,0,SNR         ;IS NEW FRAME = 0?
        JMP     ZEROF           ;YES
        STA     0,@22           ;NO, SAVE IT IN BUFF2
        ISZ     RFCNT           ;INCREMENT READER FRAME COUNTER (NEVER SKIP)
```

EXAMPLE FOUR (Continued)

```
SPTR:   INTDS               ;MAKE SURE READER STARTED AND FLAG SET TOGETHER
        NIOS  PTR           ;START READER FOR NEXT FRAME
        ISZ   RACTV         ;SET READER ACTIVE FLAG
        INTEN
        JMP   PTRDS         ;GO DISMISS INTERRUPT

ZEROF:  MOV#  3,3,SNR       ;FRAME IS ZERO, WAS PREVIOUS ONE ALSO ZERO?
        JMP   SPTR          ;YES, IGNORE LEADING BLANK FRAMES
        STA   0,@22         ;NO, DONE WITH THIS BLOCK, MARK WITH 0 FRAME
        ISZ   RFCNT         ;INCREMENT READER FRAME COUNTER
        LDA   1,BBUF2       ;AC1 POINTS TO BEGINNING OF BUFF2
        JSR   MWRIT         ;GO WRITE A BLOCK ON MAG TAPE

PTRDS:  JMP   @.+1          ;GO DISMISS INTERRUPT
        DISMS


BBUF2:  BUFF2               ;ADDRESS OF BUFF2
RFCNT:  0                   ;READER FRAME COUNTER
CHECK:  0                   ;LAST FRAME INPUT FROM READER, USED IN IGNORING
                            ; LEADING BLANK FRAMES


;       MAG TAPE SERVICE ROUTINE

MTACT:  @MTASR              ;MTA DEVICE CONTROL TABLE (PUSH BIT = 1)
        43                  ;MASKS OUT MTA, TTI, TTO

MTASR:  INTDS               ;DISABLE INTERRUPTS
        SUB   0,0           ;CLEAR MAG TAPE ACTIVE FLAG
        STA   0,MACTV
        INTEN               ;ENABLE INTERRUPTS

        DIA   0,MTA         ;READ MAG TAPE STATUS
        MOVL# 0,0,SZC       ;CHECK FOR ERRORS
        JMP   ERROR
        LDA   0,RFCNT       ;FETCH READER FRAME COUNTER
        MOV#  0,0,SZR       ;MORE WRITING TO DO?
        JMP   MOREM         ;YES
        NIOC  MTA           ;NO, CLEAR MAG TAPE
        DSZ   RBCNT         ;DECREMENT AND CHECK READER BLOCK COUNTER
        JMP   MORER         ;MORE READING TO DO
        JMP   MTADS         ;GO DISMISS INTERRUPT

MOREM:  DIB   1,MTA         ;READ ADDRESS OF NEXT WORD TO WRITE FROM BUFF2
        JSR   MWRIT         ;GO WRITE NEXT BLOCK ON MAG TAPE
        JMP   MTADS         ;GO DISMISS INTERRUPT

MORER:  INTDS               ;MAKE SURE READER STARTED AND FLAG SET TOGETHER
        NIOS  PTR           ;START READER
        ISZ   RACTV         ;SET READER ACTIVE FLAG
        INTEN
        LDA   0,ABUF2       ;REINITIALIZE BUFF2
        STA   0,22

MTADS:  JMP   @.+1          ;GO DISMISS INTERRUPT
        DISMS


ERROR:  HALTA 0             ;HALT SHOWING MAG TAPE STATUS WORD
                            ; (ERROR PROCESSING WOULD NORMALLY GO HERE)
```

EXAMPLE FOUR (Continued)

```
;         MWRIT WRITES A BLOCK OF DATA IN MEMORY ON MAG TAPE DRIVE 0
;            SIZE OF BLOCK IS -BLKSZ
;            ADDRESS OF FIRST WORD OF BLOCK IS CONTAINED IN AC1 ON ENTRY

MWRIT:    DOB    1,MTA          ;LOAD MEMORY ADDRESS COUNTER
          STA    1,LSTAD        ;SAVE LAST ADDRESS USED IN WRITING TO MAG TAPE
          LDA    1,BLKSZ        ;FETCH NEGATIVE BLOCK SIZE
          DOC    1,MTA          ;LOAD WORD COUNTER
          LDA    0,WRCOM        ;FETCH WRITE COMMAND
          INTDS                 ;MAKE SURE MAG TAPE STARTED AND FLAG SET TOGETHER
          DOAS   0,MTA          ;INITIATE WRITE OPERATION
          ISZ    MACTV          ;SET MAG TAPE ACTIVE FLAG
          INTEN
          LDA    0,RFCNT        ;NOW UPDATE READER FRAME COUNTER
          ADDZ   1,0,SNC        ;DECREASE IT BY BLOCK SIZE
          SUB    0,0            ;IF RFCNT WAS < BLKSZ, SET RFCNT TO 0
          STA    0,RFCNT        ;SAVE NEW VALUE
          JMP    0,3            ;RETURN TO CALLER

BLKSZ:    -144                  ;(NEGATIVE) BLOCK SIZE OF 100 DECIMAL
WRCOM:    50                    ;WRITE COMMAND (SELECTS DRIVE 0)
LSTAD:    0                     ;LAST ADDRESS USED IN WRITING TO MAG TAPE
ERCOM:    70                    ;ERASE COMMAND (SELECTS DRIVE 0)

;         MTAAR RESTARTS MAG TAPE AFTER POWER FAIL
;            SEVERAL INCHES OF TAPE ARE ERASED, THEN THE RECORD
;            WHICH WAS BEING WRITTEN WHEN POWER FAILED IS REWRITTEN

MTAAR:    LDA    0,ERCOM        ;ERASE SECTION OF TAPE
          DOAS   0,MTA
          SKPDN  MTA            ;WAIT TILL MTA DONE
          JMP    .-1
          LDA    0,RFCNT        ;SET RFCNT BACK TO ITS PRIOR VALUE
          LDA    1,BLKSZ
          SUB    1,0
          STA    1,RFCNT
          LDA    1,LSTAD        ;FETCH ADDRESS OF RECORD TO BE REWRITTEN
          JMP    MWRIT          ;GO WRITE RECORD (MWRIT RETURNS DIRECTLY TO
                                ;AUTO-RESTART CODE)


;         CHARACTER BUFFERS

BUFF:     .BLK   112            ;ALLOW FOR 72 CHARACTERS + CR + LF
BUFF2:    .BLK   1120           ;ALLOW FOR UP TO 74*8=592 FRAMES


;         VECTOR TABLE

VTAB:     @PFLCT                ;ADDRESS OF CONTROL TABLE FOR POWER FAIL
          SIH
          SIH
          SIH
          SIH
          SIH
          SIH
          SIH
          @TTICT                ;ADDRESS OF TTI DEVICE CONTROL TABLE
          SIH
          @PTRCT                ;ADDRESS OF PTR DEVICE CONTROL TABLE
          @PTPCT                ;ADDRESS OF PTP DEVICE CONTROL TABLE
          SIH

          SIH
          SIH
          SIH
          SIH
          SIH
```

EXAMPLE FOUR (Continued)

```
@MTACT           ;ADDRESS OF MTA DEVICE CONTROL TABLE
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH

SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH

SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH
SIH


.END  START
```

SECTION II

TERMINALS

- TELETYPES
- DGC DISPLAY 6012

This page intentionally left blank

INTRODUCTION TO TERMINALS

A terminal is a device through which the computer and its operator can interact. It has two independent parts, a keyboard, and a display or printer. The operator sends information to the computer through the keyboard, and the computer responds to the operator through the display or printer.

Characters that are transmitted between the computer and the terminal are coded in ASCII* (see Appendix C) and transmitted asynchronously. Each character is transmitted serially bit by bit between the computer and the terminal over a communications channel, at rates ranging from 110 to 4800 bits per second or "baud".

Although the terminal is two separate devices, the manner in which the devices interact with one another and the computer depends on the communications channel linking them. If the channel consists of one line from the keyboard to the computer and a second independent line from the computer to the printer or display, the two halves of the terminal can operate independently. This full use of the terminal is termed "full-duplex operation". If only one line connects the terminal to the computer, the keyboard and the computer must share this line; thus only one of the two halves of the terminal can be operating at any one time. This use of a single line channel is termed "half-duplex operation".

When a terminal operates in full-duplex, the computer must "echo" the information received from the keyboard if it wants that information to be shown on the display or printer. However, when operating in half-duplex, the keyboard transmits the information to both the computer and the display or printer.

*American Standard Code for Information Interchange.

This page intentionally left blank

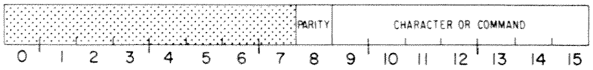
TELETYPES

SUMMARY

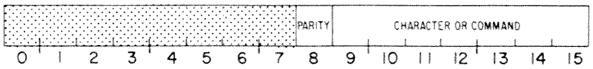
MNEMONIC (FIRST CONTROLLER)	
INPUT	TTI
OUTPUT	TTO
DEVICE CODE (FIRST CONTROLLER)	
INPUT	10_8
OUTPUT	11_8
MNEMONIC (SECOND CONTROLLER)	
INPUT	TTI1
OUTPUT	TTO1
DEVICE CODE (SECOND CONTROLLER)	
INPUT	50_8
OUTPUT	51_8
PRIORITY MASK BIT	
INPUT	14
OUTPUT	15
CHARACTERS/LINE	72
LINES/INCH	6
DATA TRANSFER RATE	
MAX (CHARACTERS/SEC)	
33 AND 35	10
37	15

ACCUMULATOR FORMATS

READ CHARACTER BUFFER........... (DIA)



LOAD CHARACTER BUFFER.......... (DOA)



S, C AND P FUNCTIONS

S	Set Busy to 1, Done to 0 and either load the Input Buffer or write a character.
C	Set both Busy and Done to 0 and terminate all data transfers. If issued before transmission is complete, partial character codes are received.
P	No effect.

INTRODUCTION

The Teletype provides for two-way communications between the computer and the operator. The keyboard is the input device and the printer is the output device. All the exchanges of data between the keyboard and the computer and between the computer and the printer utilize a subset of the 128-character alphanumeric ASCII code shown in Appendix C.

In addition to a keyboard and a printer, certain models of the Teletype terminal are equipped with a paper tape reader/punch combination. Such terminals are designated as Automatic Send/Receive (ASR) terminals. Those which are not so equipped are designated as Keyboard Send/Receive (KSR) terminals.

Three distinct Teletype models are available from Data General Corporation: the model 33, the model 35, and the model 37. Both the models 33 and 35 operate at a data transmission rate of 10 characters per second (110 baud) while the model 37 operates at 15 characters per second (150 baud).

All three terminals print up to 72 characters per line with 6 lines to an inch. Both the models 33 and 35 printers utilize 8 1/2 inch wide paper while the model 37 requires 9 1/2 inch wide paper. The 33 and 35 are upper-case only terminals while the model 37 is full upper-and lower-case. Other differences among the various models may be found in the Operator' s Reference Manual (015-000034).

INSTRUCTIONS

The following instructions and timing information are for the terminal when it is used in conjunction with a 4010 controller.

The controller contains an 8-bit Input Buffer and an independent 8-bit Output Buffer.

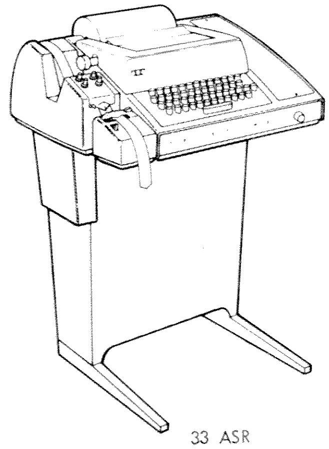
The controller's Busy and Done flags are controlled using two of the device flag commands as follows:

f = S Sets Busy to 1, Done to 0 and either reads a character into the Input Buffer or transfers the character in the Output Buffer to the printer or the punch.

f = C Sets Busy and Done to 0, thus stopping all data transfer operations. A Clear command issued during a transfer will result in the partial reception of the code being transferred.

f = P No effect.

Since the terminal is actually two devices, both a Busy and Done flag are available for input operations and a separate set of Busy and Done flags are available for output operations.



33 ASR

READ CHARACTER BUFFER

DIA<f> ac, TTI

0	1	1	AC		0	0	1	F		0	0	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Input Buffer are placed in bits 8-15 of the specified AC. Bits 0-7 of the specified AC are set to 0. After the data transfer, the controller's Input Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

▨								PARITY	CHARACTER OR COMMAND						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	----	Reserved for future use.
8	Parity	Parity bit selected at the terminal; even, odd or none.
9-15	Character	The 7-bit character or command read from the Input Buffer.

LOAD CHARACTER BUFFER

DOA<f> ac, TTO

0	1	1	AC		0	0	1	F		0	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 9-15 of the specified AC are loaded into the controller's Output Buffer. After the data transfer, the controller's Output Busy and Output Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

▨								PARITY	CHARACTER OR COMMAND						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	----	Reserved for future use.
8	Parity	Even, odd or no parity for the 7-bit code.
9-15	Character	The 7-bit character or command transmitted to the Output Buffer.

PROGRAMMING

Terminal

Since the terminal is actually two separate devices, input and output are discussed separately.

Input

Neither full- nor half-duplex input operations have to be initialized by the program. Striking a key automatically transmits the corresponding character to the controller. After the character is assembled, the Input Busy flag is set to 0, the Input Done flag is set to 1 and a program interrupt request is initiated.

The character can then be read by issuing a READ CHARACTER BUFFER instruction (DIA). The Input Done flag should then be set to 0 with either a Start or a Clear command. This allows the next character to initiate a program interrupt request when it is fully assembled.

Output

A character is loaded into the Output Buffer of the controller by issuing a LOAD CHARACTER BUFFER instruction (DOA). The character can then be transmitted to the terminal by issuing a Start command. While the character is being transmitted, the Output Busy flag is set to 1. Upon completion of the transmission, the Output Busy flag is set to 0 and the Output Done flag is set to 1, thus initiating a program interrupt request.

Each time a character is to be sent to the terminal, the Output Buffer must be reloaded with a LOAD CHARACTER BUFFER instruction. A sequence of LOAD CHARACTER BUFFER instructions together with Start commands is used to transmit a multi-character message. The program must allow each character to be transmitted before transmitting the next character.

Paper Tape

ASR Teletypes are equipped with a paper tape reader/punch. If the model is equipped with automatic reader control (TDT), the program may turn on the reader with the command DC2 and turn it off with the command DC3 (see Appendix C).

Input

When the terminal is equipped with a paper tape reader, the data input operation is similar to reading codes sent from the keyboard. A Start command causes the next eight bit code on the paper tape to be loaded into the Input Buffer. Issuing a READ CHARACTER BUFFER instruction will load the contents of the Input Buffer into the specified accumulator. The sequence of a Start command and a READ CHARACTER BUFFER instruction can be continued until the entire tape is read.

Output

Output to the paper tape punch is accomplished the same way as output to the printer is done. The characters or commands output are punched on the paper tape as well as being printed on the Teletype output paper.

TIMING

On both the model 33 and 35, a character is available in the Input Buffer for 21.59ms after Input Done is set to 1 before another character can overwrite the buffer. The corresponding time for the model 37 is 9.17ms. The difference in time is due to the fact that both the model 33 and 35 transmit 10 characters/second while the 35 can transmit up to 15 characters/second. If the paper tape reader is in use, the program has 3.41ms to issue another Start command to the reader after Input Done is set to 1 if the tape is to be kept in continuous motion.

Output timing for the printer and the paper tape punch is the same on both the model 33 and 35. The program has 4.55ms to transmit another character in order to continue printing or punching at the maximum rate. The time interval for the model 37 is 3.33ms.

CONSIDERATIONS

Input

All models ignore the parity bit in the codes received for printable characters. Both the model 33 and the model 35 also ignore the parity bit in the command codes they receive, while the model 37 will not carry out the command if the parity bit is incorrect.

When the terminal is operating in full-duplex, the program must "echo" the characters if they are to be printed at the terminal.

Half-duplex operation requires a protocol to be set up between the computer and the terminal. The protocol should be formed to resolve any conflicts over the use of the transmission line.

Output

When characters are sent to the models 33 and 35, all lower case characters are printed as their upper case equivalents.

Half-duplex operation requires a protocol to be set up between the computer and the terminal. The protocol should be formed to resolve any conflicts over transmission line use.

Since the mechanical motion initiated by a Carriage Return may not be completed before the next character is ready to be printed, some programmers issue one or two NUL characters after a Carriage Return. If this is not done, the next printable character could be displaced from its correct position.

When the last character position on a line is printed, and no format control character is sent to the terminal, all succeeding characters will overprint the last character on the line until a format control character is issued. If the program issues a CARRIAGE RETURN which is not immediately preceded or followed by a LINE FEED command, the entire line will be overprinted.

PROGRAMMING EXAMPLES

The following examples show how characters are passed among the computer, the teletype printer, the teletype keyboard, the teletype paper tape punch and the teletype paper tape reader. The first example reads a character from the Teletype keyboard, the second reads a character from the Teletype tape reader, the third prints a character on the Teletype printer and, if the punch on an ASR terminal is turned on, punches the character on the tape.

EXAMPLE 1

```
;   READ A CHARACTER FROM KEYBOARD

SKPDN    TTI       ;CHARACTER BUFFER LOADED YET?
JMP      .-1       ;NO
DIAC     1,TTI     ;READ CHARACTER AND CLEAR THE DONE FLAG
```

EXAMPLE 2

```
;   READ A CHARACTER FROM PAPER TAPE READER

NIOS     TTI       ;START READER
SKPDN    TTI       ;FRAME BUFFER LOADED YET?
JMP      .-1       ;NO
DIAC     1,TTI     ;READ FRAME AND CLEAR THE DONE FLAG
```

EXAMPLE 3

```
;   PRINT AND/OR PUNCH A CHARACTER

SKPBZ    TTO       ;PRINTER FREE?
JMP      .-1       ;NO, TRY AGAIN
DOAS     1,TTO     ;PRINT CHARACTER
```

The subroutine shown in example 4 and called by a JUMP TO SUBROUTINE instruction (JSR to TTYRD,) illustrates both reading and echoing characters on the Teletype, with Teletype interrupts disabled. It uses AC0 to store the character.

EXAMPLE 4

```
;        SUBROUTINE TO READ AND ECHO TELETYPE CHARACTERS, INTERRUPTS DISABLED

         TTYRD:  SKPDN    TTI      ;HAS CHARACTER BEEN TYPED?
                 JMP      .-1      ;NO, THEN WAIT
                 DIAC     0,TTI    ;YES, THEN READ CHARACTER AND CLEAR DONE
                 SKPBZ    TTO      ;IS TTO READY?
                 JMP      .-1      ;NO, THEN WAIT
                 DOAS     0,TTO    ;YES, THEN ECHO CHARACTER
                 JMP      0,3      ;RETURN
```

The teletype may also be programmed using the program interrupt facility. This technique may be useful in cases where a number of calculations may be performed in the time between Teletype characters. The routine shown in example 5 will read a line and echo it on the Teletype using the interrupt priority structure. It will read characters into a buffer beginning at location 1000_8. It is terminated by either a carriage return or line overflow. Line overflow is terminated by the value of MAXLL (maximum line length).

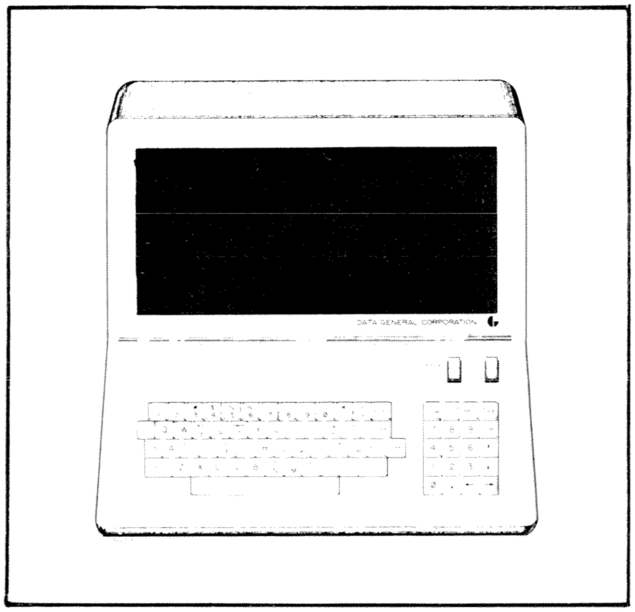
EXAMPLE 5

```
;       READ A LINE USING INTERRUPTS

        .LOC    0
        0               ;PC WILL BE STORED HERE WHEN AN INTERRUPT OCCURS
        IHAND           ;ADDRESS OF INTERRUPT HANDLER

        .LOC    400
START:  LDA     1,BUFFR ;SET UP BUFFER POINTER IN
        STA     1,23    ; AUTO-INCREMENT LOCATION 23
        LDA     1,MAXLL ;GET MAXIMUM LINE LENGTH
        STA     1,CNTR  ;INITIALIZE LINE OVERFLOW COUNTER
        SUBZL   1,1     ;SET AC1 = 1
        DOBS    1,CPU   ;MASK OUT TTO AND TURN ON INTERRUPTS
        .
        .               ;PROGRAM CAN DO USEFUL THINGS
        .               ; WHILE LINE IS BEING READ
        .
HANG:   LDA     0,CNTR  ;WHEN NEED FULL LINE TO CONTINUE,
        MOV#    0,0,SZR ; HANG UP HERE UNTIL
        JMP     .-2     ; READING IS ALL DONE
        .
        .
        .
BUFFR:  777     ;BUFFER BEGINS AT LOCATION 1000
MAXLL:  110     ;MAXIMUM OF 72 CHARACTERS PER LINE
CNTR:   0       ;LINE OVERFLOW COUNTER
        .
        .
        .
IHAND:  SKPDN   TTI     ;MAKE SURE TTI CAUSED THE INTERRUPT
        HALT            ;ERROR - SOME OTHER PERIPHERAL INTERRUPTED
        STA     0,SAV0  ;SAVE ACCUMULATORS THAT WILL BE USED
        STA     1,SAV1
        DIAC    0,TTI   ;READ CHARACTER AND CLEAR DONE
        STA     0,@23   ;STORE CHARACTER IN BUFFER
        SKPBZ   TTO     ;MAKE SURE TTO NOT BUSY
        JMP     .-1
        DOAS    0,TTO   ;ECHO CHARACTER
        LDA     1,CR    ;IS IT A CARRIAGE RETURN?
        SUB#    0,1,SZR
        JMP     .+4     ;NO
        SUBC    0,0     ;YES, CLEAR AC0 WITHOUT CHANGING CARRY
        STA     0,CNTR  ;ZERO OUT CNTR TO INDICATE LINE DONE
        JMP     .+3
        DSZ     CNTR    ;IF NOT A CARRIAGE RETURN, DECREMENT CNTR
        JMP     OUT     ;LINE NOT YET DONE, GO DISMISS
        LDA     0,TTMSK ;LINE IS DONE
        MSKO    0       ;MASK OUT TTI (AND TTO) TO INHIBIT FURTHER INPUT
OUT:    LDA     0,SAV0  ;RESTORE ACCUMULATORS
        LDA     1,SAV1
        INTEN           ;TURN INTERRUPTS BACK ON
        JMP     @0      ;RETURN TO INTERRUPTED PROGRAM

SAV0:   0
SAV1:   0
CR:     215
TTMSK:  3
```

DGC DISPLAY
6012

INTRODUCTION

The DGC Display 6012 is two separate I/O devices; a console and an alphanumeric CRT display, shown below. The console is comprised of a standard 53-station teletypewriter style keyboard, a supplementary 20-station keyboard and two switches. The first switch has three positions labeled LOCAL, OFF and ON-LINE. The ON-LINE position connects the terminal to the computer. LOCAL, used primarily for testing the display, puts the terminal off line from the computer and connects the keyboard to the display. OFF removes power from the device. The second switch has three positions labeled BUFFERED, PAGE and ROLL. Each position of this switch selects the terminal's operational mode.



The display is a 12-inch CRT with an active area of 6 by 9 inches, formatted into a twenty-four line by 80 character page. The characters that can be plotted on this screen are taken from the standard 64 character subset of ASCII, listed in Appendix C.

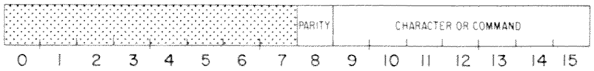
The terminal operates in three switch selectable modes called Page-buffered, Page and Roll. Page-buffered mode allows an entire page of data to be entered into the terminal's memory, edited off line and then transmitted to the computer in part or in

SUMMARY

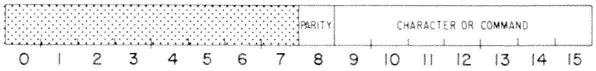
MNEMONIC (FIRST CONTROLLER)	
INPUT	TTI
OUTPUT	TTO
DEVICE CODE (FIRST CONTROLLER)	
INPUT	10_8
OUTPUT	11_8
MNEMONIC (SECOND CONTROLLER)	
INPUT	TTI1
OUTPUT	TTO1
DEVICE CODE (SECOND CONTROLLER)	
INPUT	50_8
OUTPUT	51_8
PRIORITY MASK BIT	
INPUT	14
OUTPUT	15
CHARACTERS/LINE	80
LINES/DISPLAY	24
TOTAL STORAGE CAPACITY (7-BIT CHARACTERS)	1920
DATA TRANSFER RATE MAX (BAUD)	4800

ACCUMULATOR FORMATS

READ CHARACTER BUFFER........... (DIA)



LOAD CHARACTER BUFFER.......... (DOA)



S, C AND P FUNCTIONS

S Set Busy to 1, Done to 0 and either load the Input Buffer or write a character into the display's memory.

C Set both Busy and Done to 0 and terminate all data transfers. If issued before transmission is complete, partial character codes are received.

P No effect.

whole. In this mode there are six commands for positioning the cursor, and ten additional commands for defining protected and blinking areas, setting tabs, clearing areas of memory, and transmitting characters from memory to the computer.

Page mode allows a file to be transmitted to the terminal and any desired changes made simultaneously to both the information in the terminal's memory, through the terminal hardware, and the corresponding characters stored in the computer's memory, by means of software. The commands are the same for these two modes; the only difference between them is that the keyboard in Page-buffered mode is directly coupled to the display and is off line from the computer until a special transmit key is used, while the keyboard in Page mode is always on line.

Roll mode simulates a teletypewriter. In this mode there are five commands for positioning the cursor, and three additional commands for clearing areas of memory and transmitting data. All data is entered into memory locations which correspond to the bottom line of the display screen. A LINE FEED command causes all lines on the screen to move up one, the bottom line to become blank and the top line to be lost.

INSTRUCTIONS

The following instructions and timing information are for the DGC Display 6012 when it is used in conjunction with a 4010 controller.

The 4010 controller contains an 8-bit Input Buffer and an independent 8-bit Output Buffer. Since the display is actually two devices, both a Busy and Done flag are available for input operations and a separate set of Busy and Done flags are available for output operations.

The display controller's Busy and Done flags are controlled using two of the device flag commands as follows:

f=S Sets Busy to 1, Done to 0 and either reads a character into the Input Buffer or writes the character in the Output Buffer into the display's memory.

f=C Sets Busy and Done to 0, thus stopping all data transfer operations. A Clear command issued in during a transfer will result in the partial reception of the code being transferred.

f=P No effect.

READ CHARACTER BUFFER

DIA<f> ac, TTI

0	1	1	AC		0	0	1	F		0	0	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Input Buffer are placed in bits 8-15 of the specified AC. Bits 0-7 of the specified AC are set to 0. After the data transfer, the controller's Input Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

▨	▨	▨	▨	▨	▨	▨	▨	PARITY	CHARACTER OR COMMAND						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	----	Reserved for future use.
8	Parity	Parity bit selected at the terminal; even, odd or none.
9-15	Character	The 7-bit character or command read from the Input Buffer.

LOAD CHARACTER BUFFER

DOA<f> ac, TTO

0	1	1	AC		0	1	0	F		0	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 9-15 of the specified AC are loaded into the display's Output Buffer. After the data transfer, the controller's Output Busy and Output Done flags are set according to the function specified by F. The format of the specified AC is as follows:

▨	▨	▨	▨	▨	▨	▨	▨	PARITY	CHARACTER OR COMMAND						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	----	Reserved for future use (always 0).
8	Parity	Even, odd or no parity for the 7-bit code. Ignored by the display.
9-15	Character	The 7-bit character or command transmitted to the Output Buffer.

CONTROL CHARACTERS

The control characters for the DGC Display 6012 are described in the following format:

NAME-CODE (LOCATION) D and/or ND

Functional Description

NAME:	Command Name
CODE:	Octal command code
LOCATION:	Keyboard location
D	Destructive to some information on the screen
ND	Non-destructive to information on the screen
FUNCTION:	Effect of the command

START PROTECT-36 (CTRL SH N) D

In Page-buffered and Page modes, START PROTECT is displayed as a space and is the delimiter of the beginning of a protected region. The end of the protected region is delimited by the first TAB STOP/END PROTECT character encountered, scanning from left to right and downward on the screen from the START PROTECT character. If any command moves the cursor into a protected region, the cursor will move to the first character position following the TAB STOP/END PROTECT character for that region.

Note that every START PROTECT character should have a companion TAB STOP/END PROTECT character between it and the end of the page or the entire protection mechanism is disabled in the terminal, and all START PROTECT and TAB STOP/END PROTECT characters are displayed as spaces.

In Roll mode, START PROTECT has no effect.

TAB STOP/END PROTECT-35 (CTRL SH M) D

In Page-buffered and Page modes, TAB STOP, END PROTECT is displayed as a space and does one of two things: it is either a tab stop or it is the delimiter of the end of a protected region or both. If it is a tab stop, when a TAB command is issued the cursor moves to the first character position following the first unprotected TAB STOP/END PROTECT character encountered on the screen, scanning from left to right and downward on the page from the cursor position. If it is a delimiter of the end of a protected region, then the beginning of that region must be delimited by a START PROTECT character. If any command moves the cursor into a protected region, the cursor will move to the first character position following the TAB STOP/END PROTECT character for that region.

In Roll mode, TAB STOP/END PROTECT has no effect.

HOME-10 (CTRL H) or (HOME) ND

In Page-buffered and Page modes, HOME moves the cursor to the first position in the top line on the screen. If this position is in a protected region, HOME will move the cursor to the first position following the end protect character for that region. If issued from the processor, HOME will terminate a transmission initiated by a TRANSMIT BUFFER command.

In Roll mode, HOME moves the cursor to the first position in the bottom line of the display screen. If issued from the processor, HOME will terminate a transmission initiated by a TRANSMIT BUFFER command.

CARRIAGE RETURN-15 (CTRL M) ND

In Page-buffered and Page modes, CARRIAGE RETURN moves the cursor to the first character position of the line the cursor occupies. If the first position of the line is in a protected region, the cursor will move to the first position following the TAB STOP/END PROTECT character for that region.

In Roll mode, CARRIAGE RETURN moves the cursor to the first character position in the bottom line.

LINE FEED-12 (CTRL J) or (↓) D/ND

In Page-buffered and Page modes, LINE FEED moves the cursor down the screen one line. When the cursor is in the bottom line, LINE FEED has no effect. If a LINE FEED moves the cursor into a protected region, the cursor will move to the first character position following the TAB STOP/END PROTECT character for that region.

In Roll mode, LINE FEED moves all the lines of data on the display screen up one line. The top line of the display screen is lost and the bottom line becomes blank. The cursor remains in its current position in the bottom line.

TAB-11 (CTRL I) or TAB ND

In Page-buffered and Page modes, TAB moves the cursor to the position following the TAB STOP/END PROTECT character encountered on the screen, scanning from left to right and downward on the screen. If no TAB STOP/END PROTECT character is found on the screen between the cursor position and the end of the page, the cursor moves to the first character position in the top line. If this position is in a protected region, TAB moves the cursor to the first position following the TAB STOP/END PROTECT character for that region.

In Roll mode, TAB has no effect.

CURSOR UP-17 (CTRL O) or (↑) ND

In Page-buffered and Page modes, CURSOR UP moves the cursor up one line. When the cursor is in the top line of the display screen, CURSOR UP has no effect. If the command CURSOR UP moves the cursor into a protected region, the cursor will move to the first character position following the TAB STOP/END PROTECT character for that region.

In Roll mode, CURSOR UP has no effect.

CURSOR RIGHT-30 (CTRL C) or (→) ND

In Page-buffered and Page modes, CURSOR RIGHT moves the cursor one character position to the right. When the cursor is in the last character position of the line, the cursor will move to the first character position in the next line down the page. When the cursor is in the last character position of the bottom line, CURSOR RIGHT has no effect. If the command CURSOR RIGHT moves the cursor into a protected region, the cursor will move to the first position following the TAB STOP/END PROTECT character for that region.

In Roll mode, CURSOR RIGHT moves the cursor one character position to the right. When the cursor is in the last character position of the bottom line, CURSOR RIGHT has no effect.

CURSOR LEFT-31 (CTRL Y) or (←) ND

In Page-buffered and Page modes, CURSOR LEFT moves the cursor one character position to the left. When the cursor is in the first character position of a line, the cursor will move to the last character position on the line above it. When the cursor is in the first character position of the top line, CURSOR LEFT has no effect. If the command CURSOR LEFT moves the cursor into a protected region, the cursor moves to the first position following the TAB STOP/END PROTECT character for that region.

In Roll mode, CURSOR LEFT moves the cursor one character position to the left. When the cursor is in the first character position of the bottom line, CURSOR LEFT has no effect.

CLEAR TO END OF LINE-13 (CTRL K) or (CLEAR EOL) D

In Page-buffered and Page modes, CLEAR TO END OF LINE erases all unprotected data from the cursor position to the end of the line, inclusive. The cursor does not change position.

In Roll mode, CLEAR TO END OF LINE erases all data from the cursor position to the end of the line, inclusive. The cursor does not change position.

CLEAR SCREEN-14 (CTRL L) or (CLEAR) D

In Page-buffered and Page modes, CLEAR SCREEN erases all unprotected data on the display screen. The cursor moves to the first character position of the top line. If the first position of the top line is in a protected region, CLEAR SCREEN moves the cursor to the first position following the TAB STOP/END PROTECT character for that region.

In Roll mode, CLEAR SCREEN erases all data on the screen and moves the cursor to the first character position of the bottom line. Data cannot be protected in Roll mode.

FORCE ERASE-34 (CTRL SHL) D

In Page-buffered and Page mode, FORCE ERASE erases all data on the screen, including all protected areas. The cursor moves to the first character position in the first line.

In Roll mode, FORCE ERASE has no effect.

BLINK-37 (CTRL SHO) D

In Page-buffered and Page modes, BLINK causes any character, or characters, between two BLINK characters to flicker on the display screen. If a single BLINK character is entered on the page, all characters from that position to the end of the page will flicker. BLINK characters are displayed as spaces and are transmitted as underscores. The cursor moves one character position to the right. In Roll mode, BLINK has no effect.

TRANSMIT BUFFER-16 (CTRL N) ND

In all modes, TRANSMIT BUFFER sends to the processor the contents of the terminal's memory, character by character, from the cursor position to the end of the page. Any protected regions encountered will not be transmitted. The data on the display screen will not be disturbed. TRANSMIT BUFFER moves the cursor to the last character position of the last line. If this position is protected, the cursor moves to the first unprotected position on the page. Transmission can be halted at any point by having the program issue a HOME command.

CONTROL KEYS

XMIT ND

XMIT allows the operator to transmit a message to the processor while the terminal is in Page-buffered mode. The message is sent by holding down the XMIT key while typing characters on the keyboard. If the terminal is operating in full-duplex and the program does not echo the characters back to the terminal, the data on the display screen remains undisturbed. If the terminal is operating in half-duplex or if the program echoes characters, then the message entered will overwrite data on the display screen.

In Page and Roll modes, XMIT has no effect.

BREAK ND

In all modes, while BREAK is depressed, the terminal's transmitter is disabled so that no characters are transmitted from either the keyboard or the memory.

REPEAT ND

The REPEAT key provides the continuous transmission of any code as long as both the REPEAT key and the code's corresponding key(s) are held down together.

SHIFT and CTRL ND

The SHIFT and CTRL keys produce commands or alphanumeric codes when they are depressed together with other keys.

ESC -33 (ESC) (CTRL) SHIFT K) ND

In all modes, ESC sends code 33, a protocol character. ESC does not work together with REPEAT.

CTRL RESET D

CTRL RESET clears the entire display memory, initializes the control, and places the cursor in the first position of the bottom line.

PROGRAMMING

Since the terminal is actually two separate devices, input and output are discussed separately.

Input

Neither full- nor half-duplex input operations have to be initialized by the program. Striking a key in either Page or Roll modes automatically transmits the corresponding character to the controller. After the character is assembled, the Input Busy flag is set to 0, the Input Done flag is set to 1 and a program interrupt request is initiated.

The character can then be read by issuing a READ CHARACTER BUFFER instruction (DIA). The Input Done flag should then be set to 0 with either a Start or a Clear command. This allows the next character to initiate a program interrupt request when it is fully assembled.

The TRANSMIT BUFFER command transmits the contents of the terminal's memory character by character to the controller.

Output

A character is loaded into the Output Buffer of the controller by issuing a LOAD CHARACTER BUFFER instruction (DOA). The character can then be transmitted to the terminal by issuing a Start command. While the character is being transmitted, the Output Busy flag is set to 1. Upon completion of the transmission, the Output Busy flag is set to 0 and the Output Done flag is set to 1, thus initiating a program interrupt request.

Each time a character is to be sent to the terminal, the Output Buffer must be reloaded with a LOAD CHARACTER BUFFER instruction. A sequence of LOAD CHARACTER BUFFER instructions together with Start commands is used to transmit a multi-character message. The program must allow each character to be transmitted before transmitting the next character.

TIMING

Input Timing

After the Input Done flag is set to 1, and before another key strike can destroy the character in the Input Buffer, the character is available for a READ CHARACTER BUFFER instruction for a time interval determined by the baud rate.

Time Available (ms)	Baud Rate
21.59	110
15.84	150
7.92	300
3.95	600
1.97	1200
1.31	1800
.98	2400
.65	3600
.49	4800

Output Timing

After the Output Done flag is set to 1, the program should provide another character within a time limit determined by the baud rate to keep the transmission line operating at its maximum rate.

Time Limit (ms)	Baud Rate
9.15	110
6.64	150
3.32	300
1.66	600
.83	1200
.55	1800
.42	2400
.27	3600
.21	4800

CONSIDERATIONS

The command TRANSMIT BUFFER, which is designed to be used primarily in Page-buffered mode, can also be issued when the display is in Page or Roll modes.

Input

The codes received from the terminal can be selected, at the terminal, to be 5, 6, 7, or 8 bits long with even, odd, or no parity bit. The programmer should determine the code structure used in the terminal and make sure that the controller is compatible.

When the terminal is operating in full-duplex, the program must "echo" the characters if they are to affect the display screen.

Half-duplex operation requires a protocol to be set up between the computer and the terminal. The protocol should be formed to resolve any conflicts over the use of the transmission line.

Output

The codes received by the terminal can be selected, at the terminal, to be 5, 6, 7, or 8 bits long. The parity bit is ignored in all codes received by the terminal.

When characters are sent to the terminal, all lower case characters are displayed as their uppercase equivalents.

Half-duplex operation requires a protocol to be set up between the computer and the terminal. The protocol should be formed to resolve any conflicts over transmission line use.

When operating in either Page-buffered or Page mode, characters will automatically continue to the next line when the end of the current line is reached. When the last line on the page is filled, any other characters received will overwrite the last character on the last line. When operating in Roll mode, the last character in the bottom line will be overwritten by subsequent characters. In order to avoid overwriting any line, both a CARRIAGE RETURN and a LINE FEED command should be issued.

This page intentionally left blank

SECTION III

HARD COPY

- PAPER TAPE READER
- PAPER TAPE PUNCH
- CARD READERS
- LINE PRINTERS
- PLOTTERS

This page intentionally left blank

INTRODUCTION TO HARD COPY PERIPHERALS

Hard copy peripherals are devices through which data is transferred into and out of the computer system. Data is entered into the system through input devices which read or sense coded data from paper tape or cards, and transferred out of the system to output devices which record the data on paper tape, line printer paper, or plotting paper.

Paper tape can be used as either an input or an output medium. The medium is a long strip of 1/2" wide paper or mylar tape. A series of holes located across the width of the tape represent a frame of information. Each frame contains eight bits. The information on the tape may be entered into the computer system through either a high speed paper tape reader or through a Teletype equipped with a paper tape reader. The maximum transfer rate for a high speed reader is 400 frames/second while the input rate from an ASR Teletype is either 10 or 15 frames/second, depending on the particular model used.

Information may be transferred out of the system to either a high speed paper tape punch or to a Teletype equipped with a paper tape punch. The maximum transfer rate to a high speed punch is 63.3 frames/second while the rate to an ASR Teletype is either 10 or 15 frames/second, depending on the particular model used.

Cards for input may be of two types: industry standard 12-row 80-column punched cards or 12-row variable-format mark-sense cards. A series of 12 locations across the width of a card represents a column of information which is coded as holes in the appropriate locations on a punched card or as pencil marks in the appropriate locations on mark sense cards. The information transfer rate depends on the particular model of the card reader used and the format of the card. The range is from 150 to 1000 cards/minute.

Line printers provide high speed hard copy output for alphanumeric information. The paper is generally sprocket-fed, fan-fold and of widths ranging from 4 to 19 7/8 inches. Alphanumeric information is sent to a line printer in either 64 or 96 character subsets of ASCII code. Depending on the particular model used, information can be transferred at a rate of up to 300 136-character lines/minute.

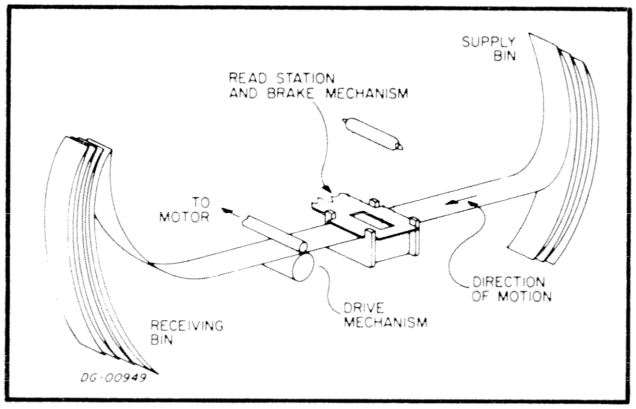
Graphical output in the form of charts and drawings is provided by the incremental plotters. Several variations allow plotting on either a single sheet, a roll, or a fanfold stack of paper. All plotters allow the selection of 8 possible line segments which can be generated in each incremental step. The length of the basic step size may be specified upon ordering to be either metric (.05 - .25mm) or english (.002 - .010 inch). Depending on the model used, the plotter draws at either 200 or 300 steps/second.

This page intentionally left blank

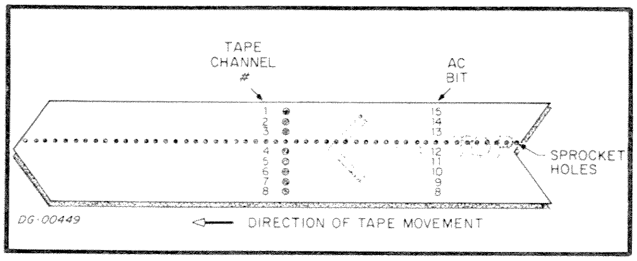
PAPER TAPE READER

INTRODUCTION

Paper tape readers provide data input from standard fanfold eight-channel paper or mylar tapes at speeds of up to 300 or 400 frames/second (4011B and 6013 readers, respectively). The reader consists of a supply bin, a read station, and a receiving bin. Tape is moved from the supply bin through the read station, where each frame is read, to the receiving bin where it may be removed.



The tape format is shown below. The eight channels across the width of the tape comprise a frame. The sprocket hole is used as a timing strobe for each frame as it enters the read station. Both the code structure used for data and the interpretation of the input is determined by conventions decided upon by the programmer. Conventional ASCII paper tape code may be found in Appendix C.



SUMMARY

MNEMONIC (FIRST CONTROLLER) ..... PTR

DEVICE CODE (FIRST CONTROLLER)..... 12_8

MNEMONIC (SECOND CONTROLLER) .. PTR1

DEVICE CODE (SECOND CONTROLLER)... 52_8

PRIORITY MASK BIT 11

BITS/FRAME 8

FRAMES/INCH 10

CAPACITY OF HOPPER (FEET) 100-150

MAXIMUM DATA TRANSFER RATE (FRAMES/SECOND) .... 300 or 400

ACCUMULATOR FORMATS

READ FRAME .... (DIA)

▨	CHAN 8	CHAN 7	CHAN 6	CHAN 5	CHAN 4	CHAN 3	CHAN 2	CHAN 1

S, C AND P FUNCTIONS

S Set the Busy flag to 1, the Done flag to 0, and load the Frame Buffer with the contents of the frame.

C Set the Busy and Done flags to 0 without affecting the contents of the Frame Buffer.

P No effect.

INSTRUCTIONS

The tape reader is driven by a controller which contains an eight-bit Frame Buffer. If a hole is punched in a channel of a frame on a tape, a 1 will be loaded into the data bit corresponding to that channel when the frame is loaded into the Frame Buffer. The sprocket hole is not loaded into the Frame Buffer, but signals the controller when a frame enters the read station.

One I/O instruction is used to program the tape reader. This instruction loads the contents of the Frame Buffer into an accumulator.

The tape reader controller's Busy and Done flags are controlled by the flag commands as follows:

f=S Set the Busy flag to 1, the Done flag to 0, and load the Frame Buffer with the contents of the next frame on the tape.

f=C Set both the Busy and Done flags to 0 without affecting the contents of the Frame Buffer.

f=P No effect.

READ FRAME

DIA<f> ac, PTR

0	1	1	AC		0	0	1	F		0	0	1	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Frame Buffer are loaded into bits 8-15 of the specified AC. Bits 0-7 are set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

								CHAN 8	CHAN 7	CHAN 6	CHAN 5	CHAN 4	CHAN 3	CHAN 2	CHAN 1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	----	Reserved for future use.
8	channel 8	A hole in the corresponding channel(s) of the frame read places a 1 in the corresponding accumulator position(s).
9	channel 7	
10	channel 6	
11	channel 5	
12	channel 4	
13	channel 3	
14	channel 2	
15	channel 1	

PROGRAMMING

Once the operator has loaded the tape into the reader and placed the reader on line, the program may read the tape. A Start command (NIOS) issued to the reader will load the Frame Buffer with the contents of the next frame on the tape. While the buffer is being loaded, the Busy flag is 1, and the Done flag is 0. When the buffer has been loaded, the Busy flag is set to 0, and the Done flag is set to 1, thus initiating a program interrupt request. The program may then read the contents of the buffer by issuing a READ FRAME instruction (DIA). The READ FRAME instruction loads the contents of the Frame Buffer into the specified accumulator. The program may continue reading frames by issuing a series of Start commands and READ FRAME instructions.

TIMING

The paper tape reader is capable of reading at speeds of up to 300 (4011B) or 400 (6013) frames per second. When operating at this speed, the reader takes 2.5 milliseconds to fill the Frame Buffer with the next frame on the tape. In order to keep the tape in continuous motion, the program must retrieve the data, and set Busy to 1 within 100 microseconds after the Done flag is set to 1. Waiting longer than this time forces the reader to stop and restart the tape. The programmer should not attempt to operate the reader in this manner at speeds in excess of 150 frames per second. Faster stop/start rates produce chatter and may lead to unreliable reader operation.

CONSIDERATIONS

Usually, the tape has a leader which is composed on a series of null frames. Since the contents of the Frame Buffer are indeterminate when the computer is first turned on, the frames on the leader may be used to set the contents of the Frame Buffer to 0. The leader may be ignored by checking each frame, as it is read into the buffer, for non-zero contents. Processing of the information field may begin when the program finds a non-zero frame. The program usually recognizes the end of the information field on the tape by some predetermined contents of a frame or group of frames which signify the end of tape. A second method for determing the end of tape is by counting the total number of frames in the information field on the tape.

PROGRAMMING EXAMPLES

To program the paper tape reader, all that is required are four instructions. An example of this, inserted in a program would be the following:

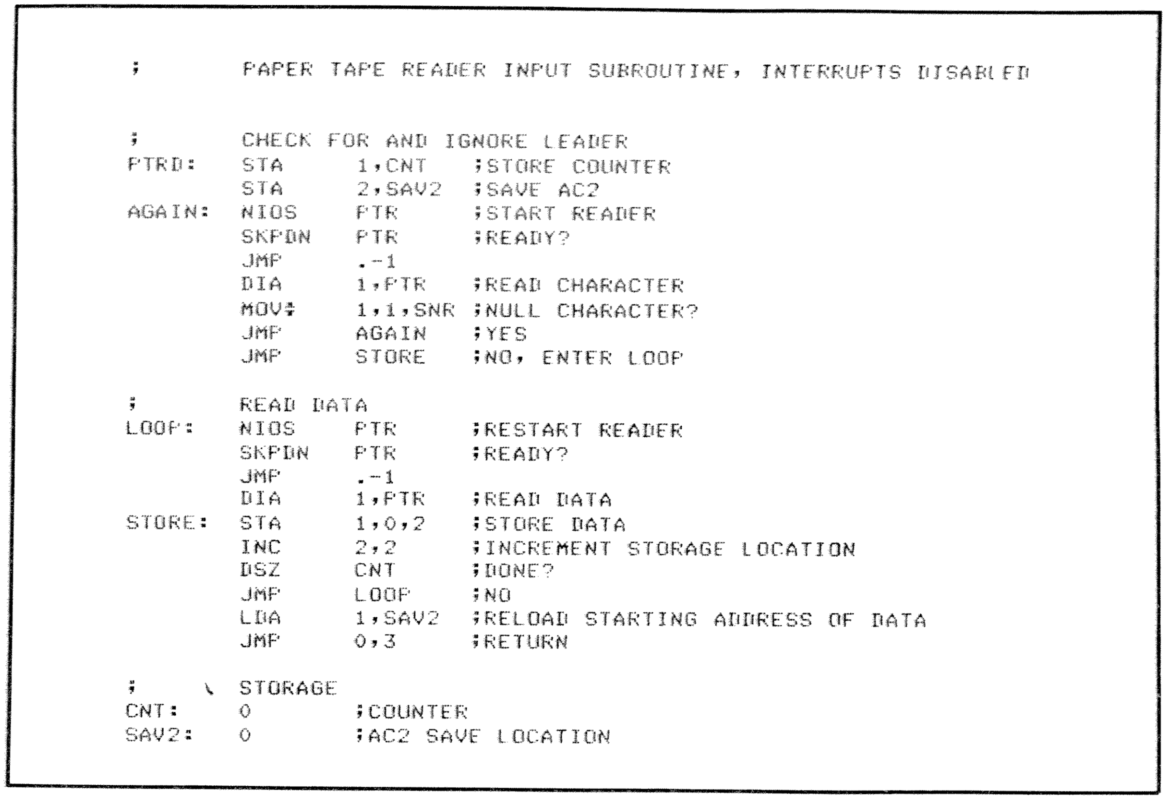
```
NIOS     PTR      ;START READER
SKPDN    PTR      ;FRAME BUFFER LOADED YET?
JMP      .-1      ;NO
DIAC     1,PTR    ;READ THE FRAME AND CLEAR THE DONE FLAG
```

But many times, more than one character may need to be read, and four simple instructions are not enough. In addition, there may be a leader (a number of blank frames at the beginning of the tape) which the programmer may want to ignore. The following subroutine reads a specified number of frames, ignoring the leader, and stores them, sequentially, starting at the address contained in AC2. When the subroutine is called, AC1 contains the number of characters to be read. Upon return to the main program, AC0 is untouched, AC1 contains the starting address of the data storage, and AC2 contains an address which is one more than the final address of the data. The subroutine is called with a JUMP TO SUBROUTINE instruction (JSR).

For example, to read 60_8 frames and store them sequentially, starting at location 4120, give:

```
          .
          LDA      1,SIXTY ;GET NUMBER OF FRAMES TO READ
          LDA      2,ADDR  ;GET BEGINNING OF BUFFER
          JSR      PTRD    ;GO TO READER SUBROUTINE
          .
          .
          .
SIXTY:    60
ADDR:     4120
```

This will call the following routine:



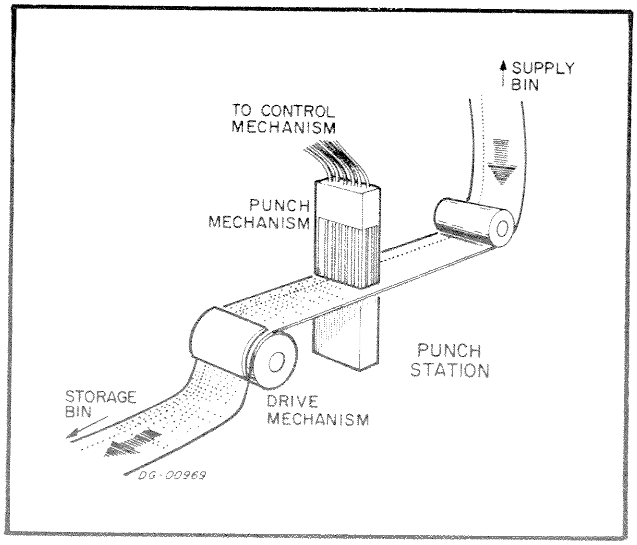
```
;       PAPER TAPE READER INPUT SUBROUTINE, INTERRUPTS DISABLED


;       CHECK FOR AND IGNORE LEADER
PTRD:   STA     1,CNT   ;STORE COUNTER
        STA     2,SAV2  ;SAVE AC2
AGAIN:  NIOS    PTR     ;START READER
        SKPDN   PTR     ;READY?
        JMP     .-1
        DIA     1,PTR   ;READ CHARACTER
        MOV#    1,1,SNR ;NULL CHARACTER?
        JMP     AGAIN   ;YES
        JMP     STORE   ;NO, ENTER LOOP

;       READ DATA
LOOP:   NIOS    PTR     ;RESTART READER
        SKPDN   PTR     ;READY?
        JMP     .-1
        DIA     1,PTR   ;READ DATA
STORE:  STA     1,0,2   ;STORE DATA
        INC     2,2     ;INCREMENT STORAGE LOCATION
        DSZ     CNT     ;DONE?
        JMP     LOOP    ;NO
        LDA     1,SAV2  ;RELOAD STARTING ADDRESS OF DATA
        JMP     0,3     ;RETURN

;     \ STORAGE
CNT:    0       ;COUNTER
SAV2:   0       ;AC2 SAVE LOCATION
```

Both of these examples are inefficient because they must wait in a loop during the 2.5 milliseconds it takes to complete loading in the Frame Buffer. If there are other devices that could be serviced while waiting, or calculations that could be performed, and interrupt service routine may be useful. An example of a paper tape reader service routine in an interrupt handler can be found in example three in Part 1 of this manual.

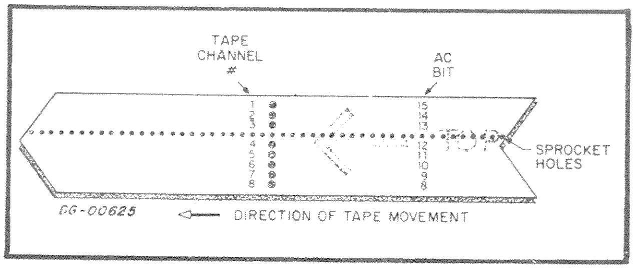
PAPER TAPE PUNCH

INTRODUCTION

The paper tape punch provides data output to standard-fanfold eight-channel paper or mular tapes. The punch consists of a supply bin, a punch station, and a take-up bin. Tape is moved from the supply bin to the punch station, where it is punched, and then to the storage bin where it may be removed. The punch can operate at speeds of up to 63.3 frames per second. Some punches are equipped with a program controlled ON/OFF switch.



The format of the tape is shown below. The eight channels across the width of the tape comprise a frame. The sprocket hole is punched to allow the tape to be read by standard paper tape readers. Both the code structure and interpretation of the input is determined by conventions decided upon the programmer. Conventional ASCII paper tape code is listed in Appendix C.



SUMMARY

MNEMONIC (FIRST CONTROLLER) .... PTP

DEVICE CODE (FIRST CONTROLLER)....... 13_8

MNEMONIC (SECOND CONTROLLER) PTP1

DEVICE CODE (SECOND CONTROLLER)..... 53_8

PRIORITY MASK BIT 13

BITS/FRAME 8

FRAMES/INCH 10

MAXIMUM TAPE LENGTH
IN STORAGE HOPPER (FEET)...........300

MAXIMUM DATA TRANSFER RATE
(FRAMES/SECOND) 63.3

ACCUMULATOR FORMAT

LOAD FRAME BUFFER................ (DOA)

								CHAN 8	CHAN 7	CHAN 6	CHAN 5	CHAN 4	CHAN 3	CHAN 2	CHAN 1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

S, C AND P FUNCTIONS

S Set the Busy flag to 1, the Done flag to 0, punch the character in the Frame Buffer and advance the tape 1 frame.

C Set the Busy and Done flags to 0 without affecting the contents of the Frame Buffer.

P No effect.

INSTRUCTIONS

The tape punch is driven by a controller containing an eight-bit Frame Buffer. If a hole is to be punched in a channel of a frame on the tape, a 1 should be loaded into the Frame Buffer position corresponding to that channel when the buffer is loaded. The sprocket hole is not loaded into the Frame Buffer, but is punched automatically.

One I/O instruction is used to program the paper tape punch. This instruction loads the contents of an accumulator into the Frame Buffer of the controller.

The paper tape punch controller's Busy and Done flags are set according to the device flag commands as follows:

f=S Set the Busy flag to 1, the Done flag to 0, and punch the contents of the Frame Buffer on the tape. If the automatic ON/OFF option is installed, and the power switch is off, a Start command also turns on the punch. It will not affect the contents of the Frame Buffer.

f=C Set both the Busy and Done flags to 0 without affecting the contents of the Frame Buffer.

f=P No effect.

LOAD FRAME BUFFER

DOA<f> ac,PTP

0	1	1	AC		0	1	0	F		0	0	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 8-15 of the specified AC are loaded into the Frame Buffer. Bits 0-7 are ignored. After the data transfer, the punch controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

▨	▨	▨	▨	▨	▨	▨	▨	CHAN 8	CHAN 7	CHAN 6	CHAN 5	CHAN 4	CHAN 3	CHAN 2	CHAN 1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	----	Reserved for future use.
8	channel 8	A 1 is placed in the bit position(s) corresponding to the tape channel(s) in which a hole is to be punched.
9	channel 7	
10	channel 6	
11	channel 5	
12	channel 4	
13	channel 3	
14	channel 2	
15	channel 1	

PROGRAMMING

Once the operator has loaded tape into the punch, and placed the punch on-line, the program may punch the tape. Since the contents of the Frame Buffer are indeterminate when the computer is first turned on, the program should load the Frame Buffer using a LOAD FRAME BUFFER instruction (DOA), before issuing a Start command. Once the FRAME BUFFER has been loaded, a Start command should be issued to punch the frame. A Start command sets the Busy flag to 1, the Done flag to 0, and then punches the contents of the Frame Buffer. Once the frame has been punched, the tape is advanced, the Busy flag is set to 0, and the Done flag is set to 1, thus initiating a program interrupt request. To continue punching data, a series of LOAD FRAME BUFFER instructions and Start commands should be issued.

TIMING

The paper tape punch operates on a mechanically determined cycle time of 15.8ms and is capable of punching up to 63.3 frames/second. The first Start command issued will take effect at the point 11.3ms after the cycle has started. If the cycle has already passed this point, the punch will wait until the next cycle. Once the punch has been started, the Done flag will be set to 1 at the beginning of the next cycle, thus initiating a program interrupt request. In order to operate at the maximum speed, the program must issue another Start command within 11.3ms after the Done flag has been set to 1, otherwise, the punch must wait another cycle.

The program controlled ON/OFF option requires approximately a 1-second delay before punching may begin. If the Busy flag remains 0 for more than 5 seconds, the punch will be turned off automatically.

CONSIDERATIONS

To make loading the reader and unloading the punch easier, tapes are usually punched with leaders and trailers made up of a series of blank frames. In order to do this, the Frame Buffer should be loaded with all zeroes, and a series of Start commands issued. A Start command does not destroy the contents of the Frame Buffer, so a series of Start commands may be used to repeat any character.

Many times a predetermined frame of group of frames is also punched with the leader or trailer to signify the length of the record, or the end of data, for the program reading it later.

If the punch runs out of tape, the punching operation continues and no indication is given to the program. Therefore, the operator should verify that sufficient tape is present for the data he intends to punch.

The standard punch must be left on all the time that it might be used as it otherwise will not respond to the program. With the automatic power ON/OFF option, the punch can be left off. Then if the Busy flag is set to 1 when the motor is off, punching is automatically delayed about 1 second while the motor gets up to speed. It can be assumed that the motor will remain on throughout any normal punching run, but if the Busy flag remains 0 for more than 5 seconds, the motor is turned off.

PROGRAMMING EXAMPLES

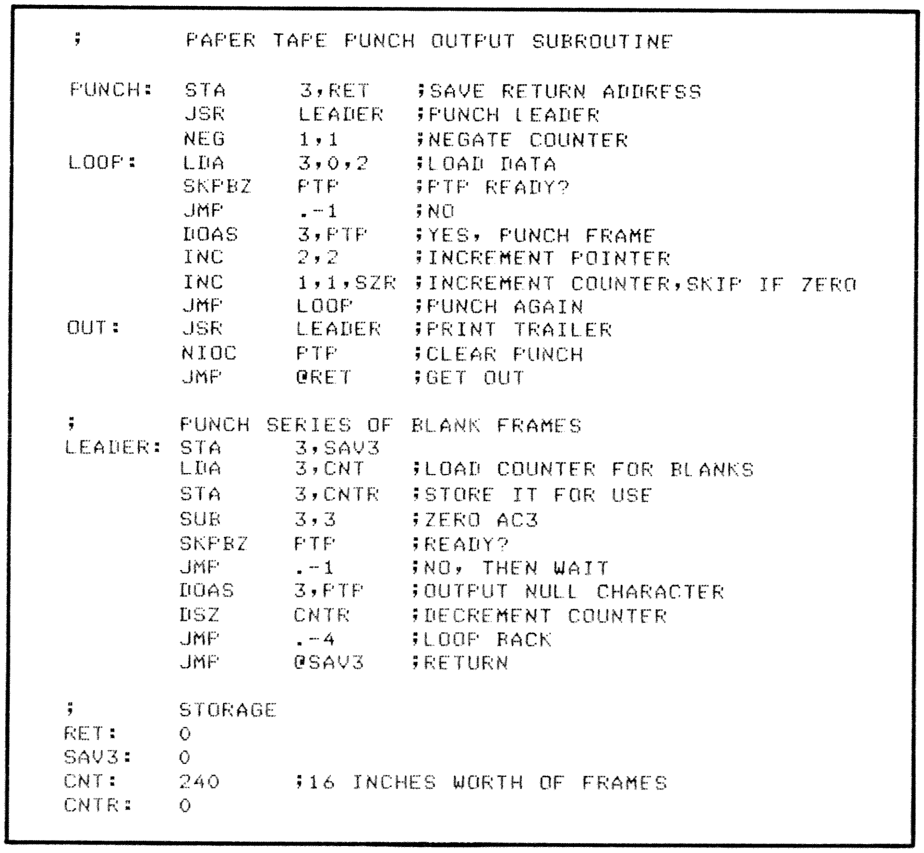
In order to punch a frame on paper tape, only three instructions are required. An example of punching a frame during a program would be:

```
SKPBZ    PTP
JMP      .-1
DOAS     1,PTP
```

It is often necessary to punch a block of frames. A subroutine to do this, including the punching of leaders and trailers would be similar to the following program. Upon calling the subroutine, AC1 contains the number of frames to be punched and AC2 contains the starting location where the data is stored sequentially, one character per word. Upon return to the main program, AC2 will contain an address one greater than the address of the last frame punched. For example, to punch 40_8 frames, stored sequentially, starting at location 1000, the following series of instructions may be used:

```
   .
   .
   .
LDA        1,           FORTY
LDA        2,           ADDR
JSR        PUNCH
   .
   .
FORTY:     40
ADDR :     1000
```

These instructions would call the following subroutine:



```
;         PAPER TAPE PUNCH OUTPUT SUBROUTINE

PUNCH:    STA       3,RET     ;SAVE RETURN ADDRESS
          JSR       LEADER    ;PUNCH LEADER
          NEG       1,1       ;NEGATE COUNTER
LOOP:     LDA       3,0,2     ;LOAD DATA
          SKPBZ     PTP       ;PTP READY?
          JMP       .-1       ;NO
          DOAS      3,PTP     ;YES, PUNCH FRAME
          INC       2,2       ;INCREMENT POINTER
          INC       1,1,SZR   ;INCREMENT COUNTER,SKIP IF 7ERO
          JMP       LOOP      ;PUNCH AGAIN
OUT:      JSR       LEADER    ;PRINT TRAILER
          NIOC      PTP       ;CLEAR PUNCH
          JMP       @RET      ;GET OUT

;         PUNCH SERIES OF BLANK FRAMES
LEADER:   STA       3,SAV3
          LDA       3,CNT     ;LOAD COUNTER FOR BLANKS
          STA       3,CNTR    ;STORE IT FOR USE
          SUB       3,3       ;ZERO AC3
          SKPBZ     PTP       ;READY?
          JMP       .-1       ;NO, THEN WAIT
          DOAS      3,PTP     ;OUTPUT NULL CHARACTER
          DSZ       CNTR      ;DECREMENT COUNTER
          JMP       .-4       ;LOOP BACK
          JMP       @SAV3     ;RETURN

;         STORAGE
RET:      0
SAV3:     0
CNT:      240       ;16 INCHES WORTH OF FRAMES
CNTR:     0
```

Both of these programs must wait in loops for the entire cycle time of the punch. In many cases, an interrupt driven paper tape punch might be desirable. An example of an interrupt driven punch may be found in the examples in part 1 of this manual.

This page intentionally left blank

CARD READERS

INTRODUCTION

Card readers provide data input from standard 12-row cards at rates up to 150-1000 cards per minute. There are two types of card readers available, one which reads standard 80-column punched cards, the other which reads optically sensed mark cards and punched cards in various 12-row formats. This second type of reader is also capable of reading inter-mixed card types, one card type at a time. The following table lists various specifications of the card readers sold by Data General Corporation.

DGC MODEL NUMBER	CARD TYPE	DATA TRANSFER RATE (CARDS/MIN)	CARD CYCLE TIME (ms)	TIME BETWEEN CHARACTERS (ms)
4016C	Punch	150	400	2.01
4016D	Punch	285	200	2.0
4016E	Punch	400	150	.87
4016F	Punch	600	100	.87
4016G	Punch	1000	60	.48
4016H	Mark Sense	150	400	161*
4016I	Mark Sense	285	210	161*
4016J	Mark Sense	400	150	69*
4016K	Mark Sense	600	100	69*
4016L	Mark Sense	1000	60	40*

*Divide this number by the number of columns/card to obtain intercharacter times.

SUMMARY

MNEMONIC (FIRST CONTROLLER)...... CDR

DEVICE CODE (FIRST CONTROLLER)...... 16_8

MNEMONIC (SECOND CONTROLLER).... CDR1

DEVICE CODE (SECOND CONTROLLER).... 56_8

PRIORITY MASK BIT 10

BITS/COLUMN 12

COLUMNS/CARD....... PUNCHED......... 80

MARK SENSE... 20-160

MAXIMUM DATA TRANSFER RATE (CARDS/MINUTE).................150-1000

ACCUMULATOR FORMATS

READ COLUMN.......................(DIA)

				ROW 12	ROW 11	ROW 0	ROW 1	ROW 2	ROW 3	ROW 4	ROW 5	ROW 6	ROW 7	ROW 8	ROW 9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

READ STATUS....................... (DIB)

											HE/SF	PICK FAIL	ERROR	READY	CARD IN READER
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

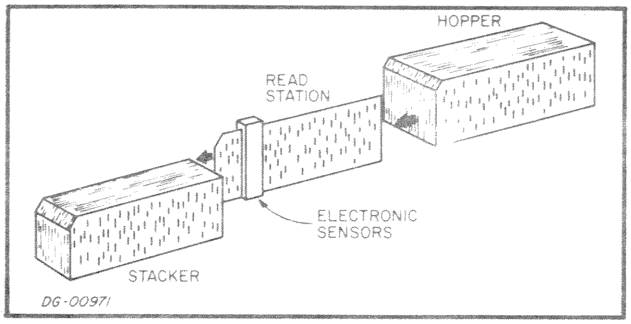
S, C AND P FUNCTIONS

S Set the Busy flag to 1, the Done flag to 0, and pick the next card.

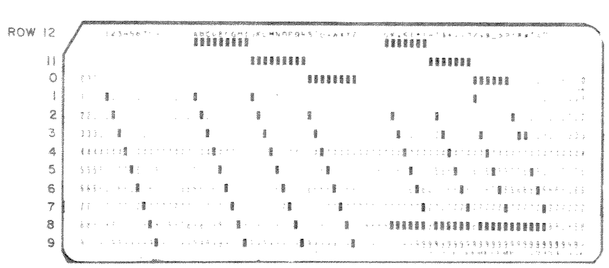
C Set both the Busy and Done flags to 0.

P Set the Done flag to 0 without affecting the Busy flag.

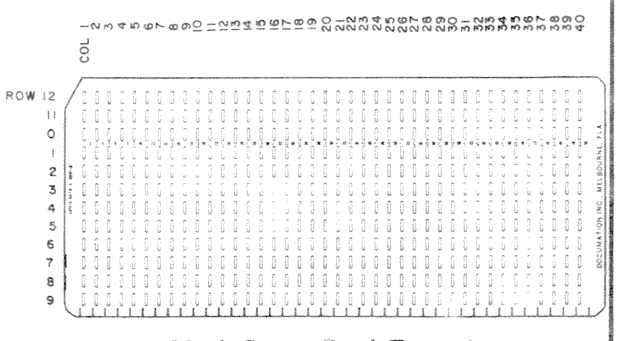
The reader is comprised of a card supply hopper, a read station, and a card stacker. A card is selected from the hopper by a mechanical pick, and is moved into the read station. Here it is read, one column at a time. Once the card has been read, it is moved to the card stacker where it may be stored temporarily, before being removed.



The formats of the cards are shown below. The twelve rows across the width of the card comprise a column. Cards are either standard 80-column, punched or variable format mark sense cards. Both the code structure used and the interpretation of data is determined by the programmer. The conventional Hollerith punched code is listed in Appendix C.



Standard 80 Column Punched Card Format


COL 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
ROW 12
11
0
1
2
3
4
5
6
7
8
9
DOCUMATION INC. MELBOURNE, FLA.

Mark Sense Card Format

INSTRUCTIONS

The card reader is driven by a controller which contains a 12-bit Data Register and a 5-bit Status Register.

Two I/O instructions are used to program the card reader; the first reads a column on the card, and the second allows the program to determine, in detail, the status of the card reader.

The card reader controller's Busy and Done flags are set according to the three device flag commands as follows:

f=S Set the Busy flag to 1, the Done flag to 0, and bring a card from the hopper into the read station.

f=C Set both the Busy and Done flags to 0. If a card is in the reader, it will continue to move through the reader, but no program interrupt requests will be generated as the columns pass through the read station.

f=P Set the Done flag to 0 without affecting the Busy flag.

READ COLUMN

DIA<f> ac,CDR

0	1	1	AC		0	1	1	F		0	0	1	1	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the controller's Data Register are loaded into bits 4-15 of the specified accumulator. Bits 0-3 are set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

				ROW 12	ROW 11	ROW 0	ROW 1	ROW 2	ROW 3	ROW 4	ROW 5	ROW 6	ROW 7	ROW 8	ROW 9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-3	----	Reserved for future use.
4	Row 12	If the row on the card is punched or marked, the corresponding bit is set to 1.
5	Row 11	
6	Row 0	
7	Row 1	
8	Row 2	
9	Row 3	
10	Row 4	
11	Row 5	
12	Row 6	
13	Row 7	
14	Row 8	
15	Row 9	

READ STATUS

DIB<f> ac,CDR

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	0	1	F		0	0	1	1	1	0

The contents of the controller's Status Register are loaded into bits 11-15 of the specified accumulator. Bits 0-10 are set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	HE/SF	PICK FAIL	ERROR	READY	CARD IN READER

Bits	Name	Meaning When Set to 1
0-10	---	Reserved for future use.
11	Hopper Empty/ Stacker Full	The input hopper has run out of cards or the output stacker is full.
12	Pick Failure	The card did not move from the input hopper into the card reader.
13	Trouble	A card is jammed in the reader or there is an electronic failure.
14	Ready	The reader is ready to take another card from the input hopper.
15	Card In Reader	A card is passing through the read station.

When the card reader is on-line, and ready to transmit data, the program may issue a Start command to the controller. A Start command sets the Busy flag to 1, the Done flag to 0, and picks a card from the hopper. Once the first column enters the read station, the Done flag is set to 1, thus initiating a program interrupt request. The Busy flag remains set to 1 as long as the card remains in the reader.

The column may then be read by a READ COLUMN instruction (DIA). A READ COLUMN instruction loads the specified accumulator with a one in those bits corresponding to punched or marked rows in a column. A Pulse command should then be issued to set the Done flag to 0 without affecting the Busy flag. This will allow the next column on the card to set the Done flag to 1 when it enters the read station, thus initiating a program interrupt request. Usually the READ COLUMN instruction and Pulse commands are combined in a DIAP instruction.

Consecutive columns may be read by a series of DIAP instructions, waiting between each for the Done flag to be set to 1.

When the final column has been read, the card passes out of the read station, the Busy flag is set to 0, and the Done flag is set to 1, thus initiating a program interrupt request. Another card may then be loaded into the reader. Note that the Done flag serves two purposes. When combined with the Busy flag being 1, the setting of the Done flag to 1 signifies a column is ready to be read. When combined with the Busy flag being 0, the setting of the Done flag to 1 signifies the end of a card.

The controller's Status Register can be checked at any time by means of a READ STATUS instruction (DIB). Before a Start command is issued, the Status Register should be checked for the Ready flag being 1, and after the Start command is issued, it should be checked for a possible pick failure. Note that if a pick failure occurs, the Done flag is not set to 1, but remains set to 0. Thus a program interrupt request will not occur. Status should also be checked after each column is read to determine any possible errors while reading. See the section entitled Error Conditions for a more detailed discussion concerning the use of the Status Register.

TIMING

The following table contains all the relevant timing information for the various card readers available from Data General Corporation. The term "card cycle time" refers to the time it takes an entire card to be loaded, processed, and put into the stacker. The term "allowable access time" refers to the amount of time the program has after the Done flag is set to 1 in order to issue a READ COLUMN instruction. If the program waits longer than this time, the data from that column is lost.

DGC MODEL NUMBER	CARD TYPE	DATA TRANSFER RATE (CARDS/MIN)	CARD CYCLE TIME (ms)	TIME BETWEEN CHARACTERS (ms)	START TO EDGE OF CARD (ms) (A)	EDGE OF CARD TO FIRST COL. (ms) (B)	START TO FIRST COLUMN (ms) (A&B)	LAST COLUMN TO END OF CARD (ms)	ALLOWABLE ACCESS TIME (ms)
4016C	Punch	150	400	2. 01	53	6. 25	59. 3	170	1. 31
4016D	Punch	285	200	2. 0	24	6. 25	30. 2	8. 05	1. 31
4016E	Punch	400	150	. 87	24	2. 6	26. 6	53. 5	. 43
4016F	Punch	600	100	. 87	24	2. 6	26. 6	3. 48	. 43
4016G	Punch	1000	60	. 48	15	1. 86	16. 9	1. 91	. 24
4016H	Mark Sense	150	400	161*	53	43. 06**	Note 1	43. 06***	. 60
4016I	Mark Sense	285	210	161*	24	43. 06**	Note 1	43. 06***	. 60
4016J	Mark Sense	400	150	69*	24	100. 67**	Note 1	100. 67**	. 23
4016K	Mark Sense	600	100	69*	24	100. 67*	Note 1	100. 67**	. 23
4016L	Mark Sense	1000	60	40*	15	175. 38**	Note 1	175. 38**	. 13

* Divide this number by the number of columns/card to obtain intercharacter times.

** Divide the distance, in inches, from the edge of the card to the second clock mark's leading edge by this factor to obtain the time interval.

*** Divide the distance, in inches, from the last column timing mark to the end of the card by this factor to obtain the time interval.

NOTE 1: Add the time calculated in column B to the time calculated in column A to obtain this time.

ERROR CONDITIONS

The card reader's Status register is used in determining when errors have been encountered during operation. These errors may cause erroneous data, or make it impossible to read data. Three status flags: Hopper Empty/Stacker Full, Trouble, and Pick Failure are available in the Status Register to signal any malfunctions in the reader. These flags will cause the reader motor to shut off and require operator intervention for correction. Two other flags: Ready, and Card-in-Reader signal the state of the reader.

If Hopper Empty/Stacker Full is set to 1, the reader will not attempt to pick another card when a Start command is issued. The Hopper Empty/Stacker Full flag indicates that either all the cards were read (Hopper Empty) or the card stacker can hold no more cards (Stacker Full). In order to continue reading cards, the operator must either load a new deck of cards, empty the card stacker, or both.

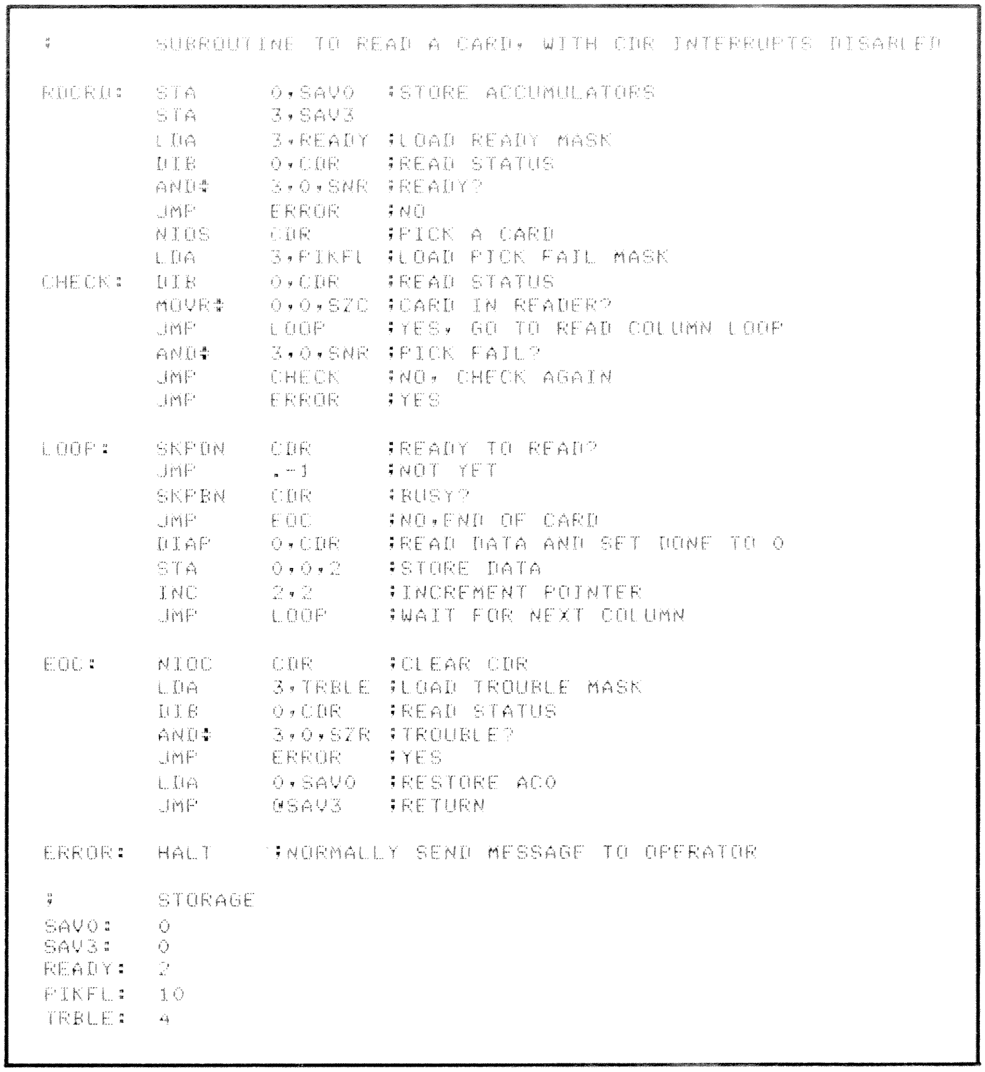
The Trouble flag indicates several types of malfunctions, such as a card jammed in the reader or a failure in the reader's electronic sensors. The reader will continue to attempt to read the card currently in the reader, but any data read from that card is questionable.

The Pick Failure flag is set to 1 if a Start command fails to bring a card into the read station from the hopper. Note that if a Pick Failure occurs, the Done flag is never set to 1, and the Busy flag remains set to 1. Therefore, the only means of determining a bad pick is through the Pick Failure flag. Even though a Pick Failure may occur, the controller will continue trying to pick the card until either a Clear command or an I/O RESET instruction (IORST) is issued. If continued attempts to read the card fail, including manual reloading by the operator, the card is probably defective and should be replaced. The Ready flag is set to 1 if the reader is ready to receive a Start command from the controller. Ready will not be 1 if any other Status flags are 1.

The Card-in-Reader flag is set to 1 if there is a card in the read station.

PROGRAMMING EXAMPLES

The card reader may be programmed in a manner similar to the Teletype input if one does not wish to check for errors in the reader. This is not recommended, as a malfunction in the reader may result in erroneous data. The following subroutine is one of many ways to read a card from the card reader, checking for errors. It is called by a JUMP TO SUBROUTINE instruction (JSR) to RDCRD. When the JSR instruction is executed, AC2 should contain the address where the data will be stored sequentially. Upon return, AC2 will contain an address one greater than the final address of the data storage. The other ACs remain unchanged. The subroutine is as follows:



```
;         SUBROUTINE TO READ A CARD, WITH CDR INTERRUPTS DISABLED

RDCRD:    STA       0,SAV0   ;STORE ACCUMULATORS
          STA       3,SAV3
          LDA       3,READY  ;LOAD READY MASK
          DIB       0,CDR    ;READ STATUS
          AND#      3,0,SNR  ;READY?
          JMP       ERROR    ;NO
          NIOS      CDR      ;PICK A CARD
          LDA       3,PIKFL  ;LOAD PICK FAIL MASK
CHECK:    DIB       0,CDR    ;READ STATUS
          MOVR#     0,0,SZC  ;CARD IN READER?
          JMP       LOOP     ;YES, GO TO READ COLUMN LOOP
          AND#      3,0,SNR  ;PICK FAIL?
          JMP       CHECK    ;NO, CHECK AGAIN
          JMP       ERROR    ;YES

LOOP:     SKPDN     CDR      ;READY TO READ?
          JMP       .-1      ;NOT YET
          SKPBN     CDR      ;BUSY?
          JMP       EOC      ;NO,END OF CARD
          DIAP      0,CDR    ;READ DATA AND SET DONE TO 0
          STA       0,0,2    ;STORE DATA
          INC       2,2      ;INCREMENT POINTER
          JMP       LOOP     ;WAIT FOR NEXT COLUMN

EOC:      NIOC      CDR      ;CLEAR CDR
          LDA       3,TRBLE  ;LOAD TROUBLE MASK
          DIB       0,CDR    ;READ STATUS
          AND#      3,0,SZR  ;TROUBLE?
          JMP       ERROR    ;YES
          LDA       0,SAV0   ;RESTORE AC0
          JMP       @SAV3    ;RETURN

ERROR:    HALT      ;NORMALLY SEND MESSAGE TO OPERATOR

;         STORAGE
SAV0:     0
SAV3:     0
READY:    2
PIKFL:    10
TRBLE:    4
```

But this program wastes time waiting for the card to be picked, and then waits for each column to be brought into the read station. The following program uses the interrupt facility to accomplish the same results with a minimum of wasted time. It is called in the same manner as the first routine, but does not return any data in the accumulators

If time is a prime concern, the pick fail check may be eliminated, releasing about 62 microseconds for other programming. But if the pick check is eliminated and a pick fail occurs, an interrupt request will never be received from the card reader.

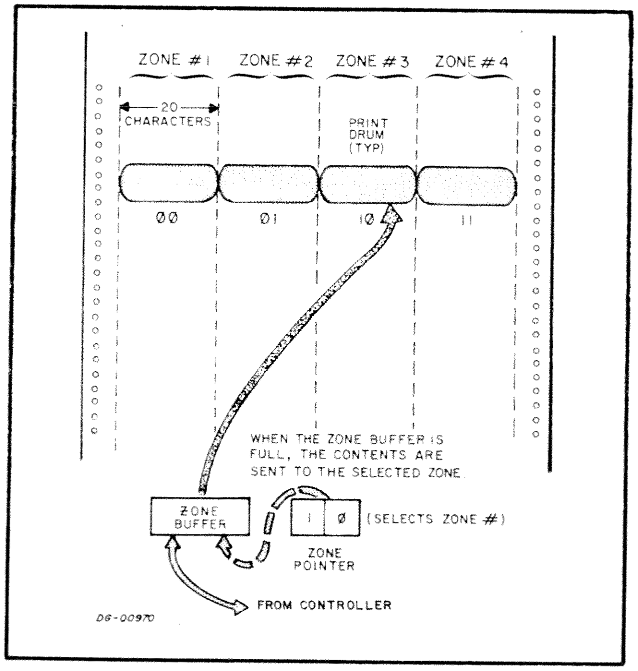
```
;       CARD READER SERVICE, USING INTERRUPTS

        .LOC    1
        INTRP

;       DUMMY INTERRUPT HANDLER
INTRP:  SKPDZ   CDR      ;CDR INTERRUPT?
        JMP     CDRSR    ;YES, GO TO CDR SERVICE ROUTINE
        HALT             ;NORMALLY WOULD CHECK FOR OTHER INTERRUPTS


;       SUBROUTINE TO READ A CARD - JSR'ED TO BY MAIN PROGRAM

RDCRD:  SKPBZ   CDR      ;READER ALREADY BUSY?
        JMP     .-1      ;YES, WAIT TILL IT'S FREE
        STA     0,SAVO   ;STORE ACCUMULATORS
        STA     3,SAV3
        STA     2,PNTR   ;STORE POINTER
        LDA     3,READY  ;LOAD READY MASK
        DIB     0,CDR    ;READ STATUS
        AND#    3,0,SNR  ;READY?
        JMP     ERROR    ;NO
        NIOS    CDR      ;PICK A CARD

;       PICK FAILURE CHECK, MAY BE OMITTED IF WAITING TIME CAN'T BE SPARED
        LDA     3,PIKFL  ;LOAD PICK FAIL MASK
CHECK:  DIB     0,CDR    ;READ STATUS
        MOVR#   0,0,SZC  ;CARD IN READER?
        JMP     LEAVE    ;YES, CARD HAS BEEN PICKED SUCCESSFULLY
        AND#    3,0,SNR  ;PICK FAIL?
        JMP     CHECK    ;NO, CHECK AGAIN
        JMP     ERROR    ;YES

LEAVE:  LDA     0,SAVO   ;RESTORE ACO
        JMP     @SAV3    ;RETURN TO MAIN PROGRAM

;       CDR INTERRUPT SERVICE ROUTINE - JMP'ED TO BY MAIN INTERRUPT HANDLER

CDRSR:  STA     0,SAVO   ;STORE ACCUMULATORS
        STA     3,SAV3
        SKPBN   CDR      ;BUSY?
        JMP     EOC      ;NO,END OF CARD
        DIAP    0,CDR    ;READ DATA AND SET DONE TO 0
        STA     0,@PNTR  ;STORE DATA
        ISZ     PNTR     ;INCREMENT POINTER
        JMP     EXIT     ;GET OUT
EOC:    NIOC    CDR      ;CLEAR CDR
        LDA     3,TRBLE  ;LOAD TROUBLE MASK
        DIB     0,CDR    ;READ STATUS
        AND#    3,0,SZR  ;TROUBLE?
        JMP     ERROR    ;YES
EXIT:   LDA     0,SAVO   ;RESTORE ACCUMULATORS
        LDA     3,SAV3
        INTEN            ;ENABLE INTERRUPTS
        JMP     @0       ;RETURN TO INTERRUPTED PROGRAM

ERROR:  HALT             ;NORMALLY WOULD SEND THE OPERATOR A MESSAGE

;       STORAGE
SAVO:   0




SAV3:   0
READY:  2
PIKFL:  10
TRBLE:  4
PNTR:   0
```

LINE PRINTERS

INTRODUCTION

Line printers provide high speed, alphanumeric, hard-copy output. Two major types of line printers are available; one using character drum impact, and the other using serial-dot matrix impact as the means of printing. The printers can receive and print one of the 64 or 96 character subsets of ASCII code shown in Appendix C.

To operate more efficiently, each line printer simultaneously prints a group of characters. In order to accomplish this, the line is divided into one or more "zones" for printing. As characters are sent to the line printer, they are stored in an area known as the "Zone Buffer". When the Zone Buffer has been filled, or the proper control character has been given, the contents of the Zone Buffer are printed in the present zone. In this manner, fewer mechanical cycles are necessary to print each line. When printing has been completed, the Zone Buffer is zeroed, and a "Zone



SUMMARY

MNEMONIC (FIRST CONTROLLER)...... LPT

DEVICE CODE (FIRST CONTROLLER)..... 17_8

MNEMONIC (SECOND CONTROLLER)... LPT1

DEVICE CODE (SECOND CONTROLLER)... 57_8

PRIORITY MASK BIT..................... 12

MAXIMUM CHARACTERS/LINE... 80/132/136

LINES/INCH........................... 6/8

ACCUMULATOR FORMATS

LOAD CHARACTER BUFFER .... (DOA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	VFU	CHARACTER OR COMMAND						

READ STATUS .... (DIA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	READY

S, C AND P FUNCTIONS

S	Set the Busy flag to 1, the Done flag to 0, and load the contents of the Character Buffer into the printer's Zone Buffer.
C	Set both the Busy and Done flags to 0 without affecting the contents of the Character Buffer.
P	No effect.

Pointer" is automatically set to point to the next zone. The Zone Pointer determines the zone to be printed next. If the zone printed was the last zone on the line, or the proper control character had been given, then the Zone Pointer returns to the first zone on the line. The program may again load the Zone Buffer for printing.

The printers are all programmed in a similar manner and incorporate the following control characters: carriage return, line feed, and form feed. Paper widths may range from 4 to 19 7/8 inches, depending upon the printer. The table below lists the general parameters for the line printers offered by Data General Corporation.

Line Printer Specifications

Model	Type	Paper Width (inches)	Full Line Print Rate	Full Line Length	Zone Size	Number of Zones
4034A	Character Drum	4 to 19 7/8	256 Lines/Min	80	20	4
4034B	Character Drum	4 to 19 7/8	245 Lines/Min	132	24	5/1/2
4034C	Serial-Dot Matrix 5x7	4 to 14 7/8	165 Characters/Sec	132	132	1
4034D	Serial-Dot Matrix 7x9	4 to 14 7/8	165 Characters/Sec	132	132	1
4034G	Character Drum	4 to 16 3/4	300 Lines/Min	136	136	1
4034H	Character Drum (Upper + Lower Case)	4 to 16 3/4	240 Lines/Min	136	136	1

DG-00972a

INSTRUCTIONS

The line printer is driven by a controller containing a 7-bit Character Buffer and a 1-bit Status Register.

Two I/O instructions are used to program the line printer. The first of these is used to transmit characters to the printer and the second is used to determine the status of the printer.

The line printer controller's Busy and Done flags are controlled by the device flag commands as follows:

f=S Set the Busy flag to 1 and the Done flag to 0. The contents of the Character Buffer in the controller are transferred to the Zone Buffer of the printer. If a Start command is issued when the Zone Buffer is one character short of being full, or the last code loaded into the Zone Buffer is a control character which initiates printing, the Zone Buffer will be printed. On models 4034G and H, only a control character will initiate printing.

f=C Set the Busy and Done flags to 0 without affecting the contents of either the Character Buffer or the Zone Buffer.

f=P No effect.

LOAD CHARACTER BUFFER

DOA<f> ac,LPT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	0	F		0	0	1	1	1	1

Bits 9-15 of the specified AC are loaded into the controller's Character Buffer. On serial-dot-matrix printers, bit 8 is also loaded for vertical formatting. Bits 0-7 are ignored. The controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	VFU	CHARACTER OR COMMAND						

Bits	Name	Contents
0-7	----	Reserved for future use.
8	Vertical Format	On serial-dot matrix printers will cause vertical tabbing when set to 1.
9-15	Character	The ASCII code for the character to be transmitted.

READ STATUS

DIA<f> ac,LPT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	0	1	F		0	0	1	1	1	1

The contents of the controller's Status Register are placed into bit 15 of the specified AC. Bits 0-14 are set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	READY

Bits	Name	Contents
0-14	----	Reserved for future use.
15	Ready	The line printer is ready to receive a character.

PROGRAMMING

Line printers are output only devices with the capability of sending status information to the program. They are programmed similarly to Teletype output in that output is one character at a time. The difference lies in the fact that line printers contain a "Zone Buffer" in which all characters are stored before printing.

Output is done through a Character Buffer in the controller. A LOAD CHARACTER BUFFER instruction (DOA) loads the contents of an accumulator into the Character Buffer of the controller. A Start command loads the contents of the Character Buffer into the Zone Buffer of the line printer. When a Start command is issued, the Busy flag is set to 1, and the Done flag is set to 0 while the contents of the character Buffer are being transferred to the Zone Buffer. Upon completion of the transfer, the Busy flag is set to 0, while the Done flag remains 0.

The contents of the Zone Buffer will be printed under the following conditions:

Model 4034A and B - The last character transferred to the Zone Buffer either filled the zone or was one of the control characters: Carriage Return, Line Feed, or Form Feed.

Model 4034C and D - The last character transferred to the Zone Buffer either filled the Zone Buffer or was a Carriage Return.

Model 4034G and H - The last character transferred to the Zone Buffer was one of the control characters: Carriage Return, Line Feed, or Form Feed.

While printing, the Busy flag is set to 1 and the Done flag is set to 0. Upon completion of the print cycle, the Busy flag is set to 0 and the Done flag is set to 1, thus initiating a program interrupt request. If the zone printed was the last zone on the line, a Carriage Return is automatically executed, except on models 4034G and H.

If a Carriage Return is executed either automatically or by a command received from the program, the line will be overwritten by the next line printed, unless the Line Feed command is issued. Line overprinting may be desired in order to obtain special characters, e.g. ≠ is obtained by overprinting a slash and an equal sign.

Formatting of the output is accomplished as follows:

> Horizontal tabbing for both types of printers is done by loading the Character Buffer with a space and issuing a Start command for every column to be spaced.
>
> Vertical tabbing operations are accomplished by loading the Character Buffer with a Line Feed command, and then issuing a Start command for every line to be spaced. Upon completion of the operation, the Busy flag is set to 0 and the Done flag is set to 1, thus initiating a program interrupt request. The serial-dot matrix printers can also perform vertical tabbing operations under the control of a vertical format tape. Tabs are placed at intervals along the tape by punching holes in the tape. Vertical formatting is initiated when the Character Buffer contains a 1 in bit 8 and is loaded into the Zone Buffer.

The serial-dot matrix printers also have a Delete command which clears all the characters previously entered into the Zone Buffer.

The line printer has a 1-bit Status Register which may be checked to determine if the printer is ready to receive data. In order to accomplish this, a READ STATUS instruction (DIA) is used. A READ STATUS instruction loads the contents of the Status Register into the specified accumulator. Status should be checked before attempting to operate the printer.

CONTROL CHARACTERS

Three Control Character Functions are available for all the line printers; Carriage Return, Line Feed, and Form Feed. The actual effects of these commands are as follows:

Carriage Return $<015_8>$

The present contents of the Zone Buffer are printed; when the next printable character is received, it will be placed in the leftmost position of the first zone.

Line Feed $<012_8>$

a) Character Drum Printers

The present contents of the Zone Buffer are printed, and the paper is spaced one line. When the next printable character is received, it will be placed in the leftmost position of the first zone.

b) Serial-Dot Matrix Printers

The paper is spaced one line. The Zone Buffer is not printed and the contents remain untouched. The next printable character received will be placed in the next position in the Zone Buffer.

Form Feed $<014_8>$

a) Character Drum Printers

The present contents of the Zone Buffer are printed, and the paper is spaced to the top of the next form. When the next printable character is received, it will be placed in the leftmost position of the first zone.

b) Serial-Dot Matrix Printers

The paper is advanced to the top of the next form. The contents of the Zone Buffer are left untouched. When the next printable character is received, it is placed in the next position in the Zone Buffer.

The following four control character functions are available on the Serial-Dot Matrix Printers in addition to the three commands listed above:

Vertical Tab $<2xx_8>$

Moves the paper until the next hole is reached on the paper control ribbon. The Zone Buffer is also loaded with the character denoted by xx. The contents of the Zone Buffer remain unchanged.

Delete $<177_8>$

All characters present in the Zone Buffer are deleted. The next suitable character received will be placed in the leftmost position of the present zone.

Expanded Characters $<016_8>$

Prints the characters double size in the horizontal axis so that instead of printing 132 characters per line, only 66 characters may be printed on a line. This function may be selected at any time prior to the Carriage Return in any line. If more than 66 characters have been entered, the excess characters are deleted.

Bell $<007_8>$

Generates a 2 second audible tone in the speaker at the rear of the printer. The contents of the Zone Buffer remain unchanged.

The following two commands are implemented on the model 4034D Serial-Dot Matrix Printer.

Select $<021_8>$

Allows the printer to receive data. This is the same as activating the printer select switch.

De-Select $<023_8>$

Places the printer off-line; this is the same as deactivating the select switch.

TIMING

There are two timing cycles for the programmer to be concerned with when outputting to the line printer. These are character cycle time and print cycle time. The character cycle time is the time it takes to load a character into the Zone Buffer. This time is either 2, 6, or 12 microseconds, depending upon the type of printer. Print cycle time is the time it takesa to print a zone. Print cycle times vary from printer to printer, and depend upon the number of characters being printed. But, print cycle times are long enough to warrant the use of an interrupt handler for the line printer.

In an interrupt routine, characters should be output to the Zone Buffer successively. When the Zone Buffer is printing, the routine should return control to the main program. To test for the zone being printed, the program should wait the character cycle time, after the output, and then test for Busy being 1. If the Busy flag is 1, then the printer has entered the print cycle, and the routine should return control to the main program awaiting an interrupt.

To print at the maximum rate, the line printer's Zone Buffer should be completely loaded within 200 microseconds after the end of a print cycle. For timing characteristics of the various line printers, see the table of line printer specifications.

Line Printer Specifications

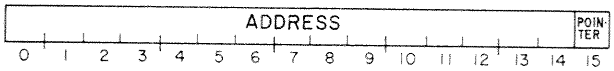
Model	Type	Paper Width (inches)	Full Line Print Rate	Full Line Length	Zone Size	Number of Zones	Character Cycle Time (μs)	Print Cycle Time	Line Feed (ms)	Carriage Return (ms)	Slew Speed
4034A	Character Drum	4 to 19 7/8	256 Lines/Min	80	20	4	6	34 ms/Zone	20	4	20 ms/Line
4034B	Character Drum	4 to 19 7/8	245 Lines/Min	132	24	5 1/2	6	34 ms/Zone	20	4	20 ms/Line
4034C	Serial-Dot Matrix 5x7	4 to 14 7/8	165 Characters/ Sec	132	132	1	12	5.5 ms/Character	70	2.75	70 ms/Line
4034D	Serial-Dot Matrix 7x9	4 to 14 7/8	165 Characters/ Sec	132	132	1	12	5.5 ms/Character	70	2.75	70 ms/Line
4034G	Character Drum	4 to 16 3/4	300 Lines/Min	136	136	1	2	50 ms/Zone	50	--	8.3 ms/Line
4034H	Character Drum (Upper+Lower Case)	4 to 16 3/4	240 Lines/Min	136	136	1	2	50 ms/Zone	50	--	8.3 ms/Line

DG-00972

PROGRAMMING EXAMPLE

The line printer may be programmed efficiently only through the use of an interrupt service routine. If interrupts are disabled, then the program must wait the entire print cycle time before it may continue processing. The following routine demonstrates a procedure for programming the line printer using interrupts.

To initiate printing, a JUMP TO SUBROUTINE instruction (JSR) to PRINT is used, where AC0 contains the starting location of the data in the form of a byte pointer (format below).

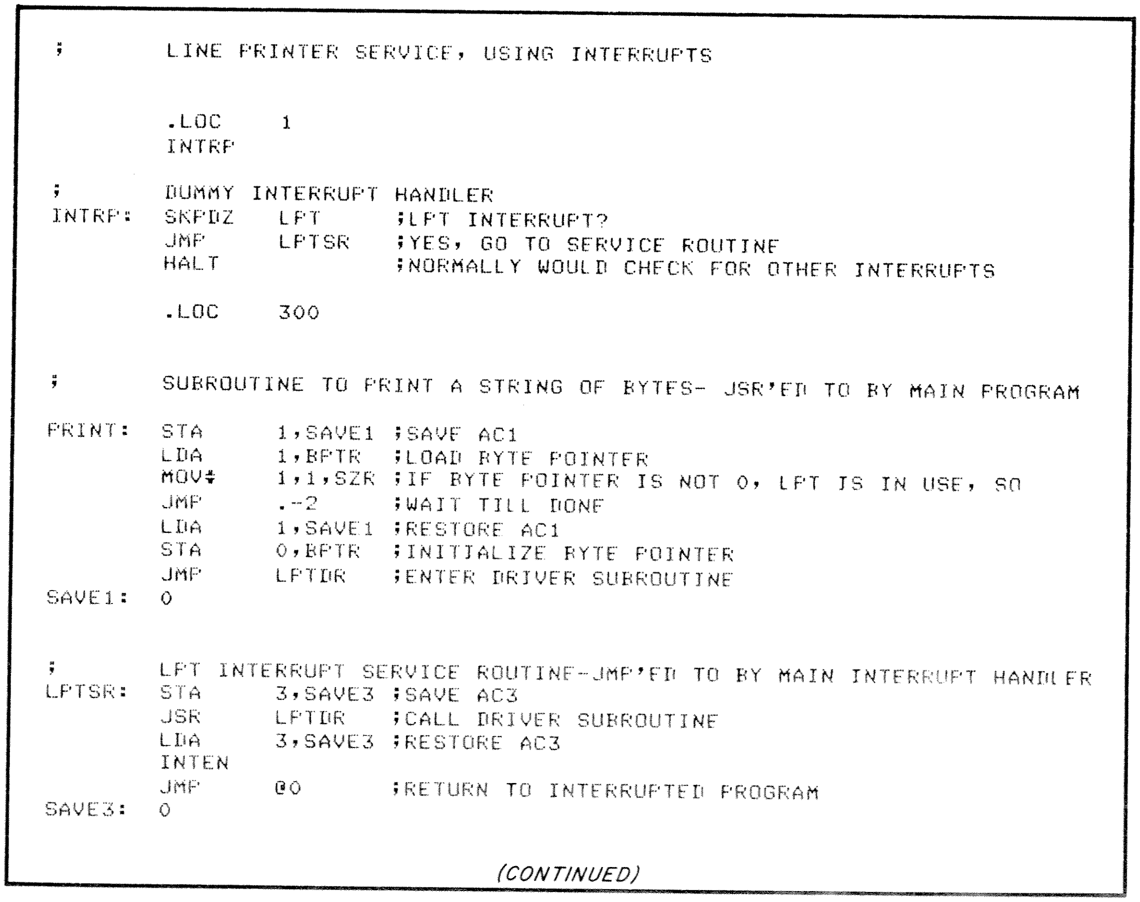


Bits	Name	Contents
0-14	Address	Address of first word of output data.
15	Pointer	Left or right byte of word 0 = left, 1 = right

The routine loads the Zone Buffer, character by character, and when the zone begins to print, it returns to the main program. It tests for the beginning of the print cycle by waiting for one character cycle time in a timing loop after each character is loaded in the Zone Buffer, and then testing the Busy flag. If the Busy flag is 1, the print cycle has begun and the routine returns control to the main program.

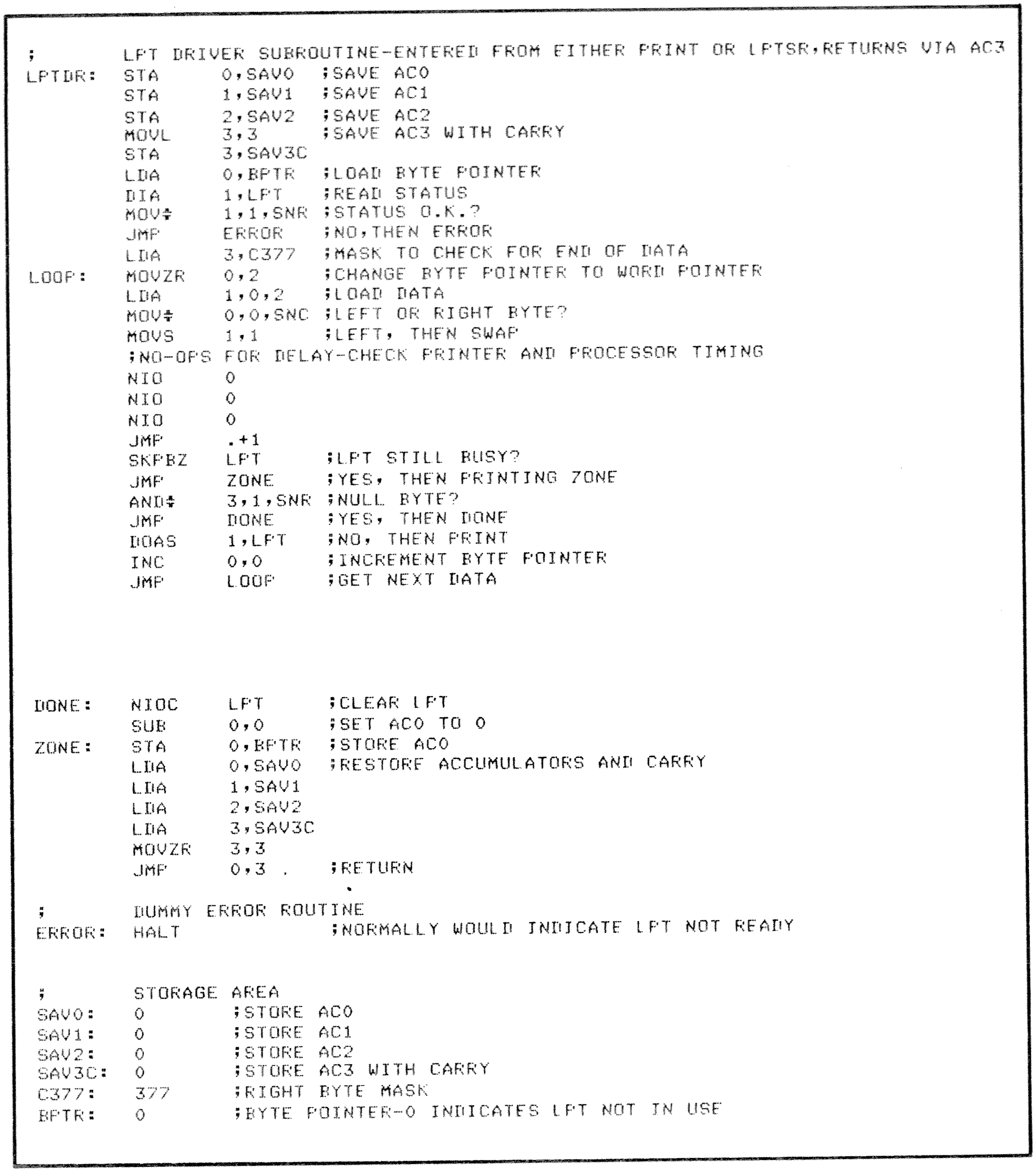
Data is packed two characters per word, left to right, and the end of data is signified by a null character (000_8). The routine sets the byte pointer to 0 when done to signify that it is not in use, and may be called by another JSR to PRINT.

If this routine is called again before it has printed the entire file, the program waits until the file is completed.



```
;         LINE PRINTER SERVICE, USING INTERRUPTS

          .LOC      1
          INTRP

;         DUMMY INTERRUPT HANDLER
INTRP:    SKPDZ     LPT       ;LPT INTERRUPT?
          JMP       LPTSR     ;YES, GO TO SERVICE ROUTINE
          HALT                ;NORMALLY WOULD CHECK FOR OTHER INTERRUPTS

          .LOC      300

;         SUBROUTINE TO PRINT A STRING OF BYTES- JSR'ED TO BY MAIN PROGRAM

PRINT:    STA       1,SAVE1   ;SAVE AC1
          LDA       1,BPTR    ;LOAD BYTE POINTER
          MOV#      1,1,SZR   ;IF BYTE POINTER IS NOT 0, LPT IS IN USE, SO
          JMP       .-2       ;WAIT TILL DONE
          LDA       1,SAVE1   ;RESTORE AC1
          STA       0,BPTR    ;INITIALIZE BYTE POINTER
          JMP       LPTDR     ;ENTER DRIVER SUBROUTINE
SAVE1:    0

;         LPT INTERRUPT SERVICE ROUTINE-JMP'ED TO BY MAIN INTERRUPT HANDLER
LPTSR:    STA       3,SAVE3   ;SAVE AC3
          JSR       LPTDR     ;CALL DRIVER SUBROUTINE
          LDA       3,SAVE3   ;RESTORE AC3
          INTEN
          JMP       @0        ;RETURN TO INTERRUPTED PROGRAM
SAVE3:    0
```

(CONTINUED)

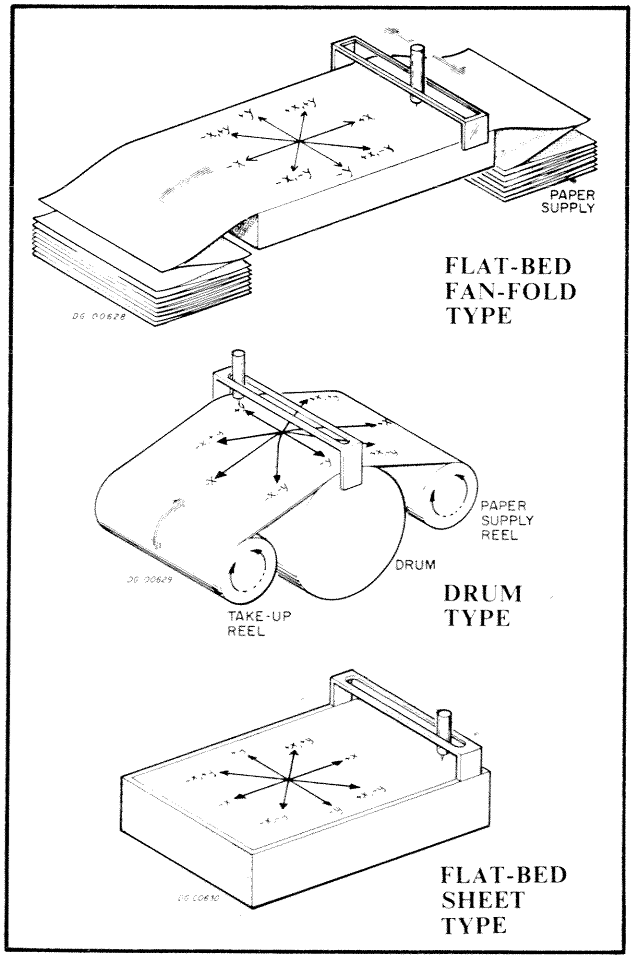


```
;       LPT DRIVER SUBROUTINE-ENTERED FROM EITHER PRINT OR LPTSR,RETURNS VIA AC3
LPTDR:  STA     0,SAV0  ;SAVE AC0
        STA     1,SAV1  ;SAVE AC1
        STA     2,SAV2  ;SAVE AC2
        MOVL    3,3     ;SAVE AC3 WITH CARRY
        STA     3,SAV3C
        LDA     0,BPTR  ;LOAD BYTE POINTER
        DIA     1,LPT   ;READ STATUS
        MOV#    1,1,SNR ;STATUS O.K.?
        JMP     ERROR   ;NO,THEN ERROR
        LDA     3,C377  ;MASK TO CHECK FOR END OF DATA
LOOP:   MOVZR   0,2     ;CHANGE BYTE POINTER TO WORD POINTER
        LDA     1,0,2   ;LOAD DATA
        MOV#    0,0,SNC ;LEFT OR RIGHT BYTE?
        MOVS    1,1     ;LEFT, THEN SWAP
        ;NO-OPS FOR DELAY-CHECK PRINTER AND PROCESSOR TIMING
        NIO     0
        NIO     0
        NIO     0
        JMP     .+1
        SKPBZ   LPT     ;LPT STILL BUSY?
        JMP     ZONE    ;YES, THEN PRINTING ZONE
        AND#    3,1,SNR ;NULL BYTE?
        JMP     DONE    ;YES, THEN DONE
        DOAS    1,LPT   ;NO, THEN PRINT
        INC     0,0     ;INCREMENT BYTE POINTER
        JMP     LOOP    ;GET NEXT DATA




DONE:   NIOC    LPT     ;CLEAR LPT
        SUB     0,0     ;SET AC0 TO 0
ZONE:   STA     0,BPTR  ;STORE AC0
        LDA     0,SAV0  ;RESTORE ACCUMULATORS AND CARRY
        LDA     1,SAV1
        LDA     2,SAV2
        LDA     3,SAV3C
        MOVZR   3,3
        JMP     0,3 .   ;RETURN

;       DUMMY ERROR ROUTINE
ERROR:  HALT            ;NORMALLY WOULD INDICATE LPT NOT READY

;       STORAGE AREA
SAV0:   0       ;STORE AC0
SAV1:   0       ;STORE AC1
SAV2:   0       ;STORE AC2
SAV3C:  0       ;STORE AC3 WITH CARRY
C377:   377     ;RIGHT BYTE MASK
BPTR:   0       ;BYTE POINTER-0 INDICATES LPT NOT IN USE
```

INCREMENTAL PLOTTERS

INTRODUCTION

Incremental plotters provide hard-copy graphical output from computer-supplied data. The plotters have three independent functions which allow ten distinct pen and/or paper movements. The three functions are x-axis movement, y-axis movement, and raise/lower pen. Eight pen and/or paper movements are possible by combining x- and y-axis commands as shown in the figure below.



SUMMARY

MNEMONIC (FIRST CONTROLLER)	PLT
DEVICE CODE (FIRST CONTROLLER)	15_8
MNEMONIC (SECOND CONTROLLER)	PLT1
DEVICE CODE (SECOND CONTROLLER)	55_8
PRIORITY MASK BIT	12
MAXIMUM PLOT SIZE (INCHES)*	31x34; 30x1440; 11x1734
STEP SIZE* (INCHES)	.002-.01
(MM)	. 05-.25
MAXIMUM RATE* (STEPS/SECOND)	200, 300

ACCUMULATOR FORMAT

LOAD COMMAND BUFFER .... (DOA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	RAISE PEN	LOWER PEN	-Y	+Y	-X	+X

S, C AND P FUNCTIONS

S Set the Busy flag to 1, the Done flag to 0, and initiate the command.

C Set both the Busy and Done flags to 0 without affecting the contents of the Command Buffer.

P No effect.

*See plotter specification.

The combination of an x- and y-axis movement generates a line 1.414 times the length of the plotter's basic step size. The pen may also be raised or lowered, providing two more commands. Each x- or y-axis command generates one step when sent to the plotter. Any directional movement which is not included in the eight basic directions must be approximated by sequential combinations of the basic steps. The table below contains information on the basic step sizes and stepping rates for the various types of plotters available from Data General Corporation.

Plotter Specifications

Plotter	Type	Paper Size (inches)	Step	Size*	Step/ Sec
4017A	Drum	12x1440 Roll	.01"	.005"	300
4017C	Drum	30x1440 Roll	.01" or .01mm		
4017D	Flat-bed	31x34 Sheet	.01" .002" or .05mm	.005" .10mm	300
4017E	Flat-bed	11x1734 Fan-fold	.01" .25mm	.005" .10mm	300

*Each plotter must be ordered with a designated step size.

Two major types of plotters are available: flat-bed and drum. Flat-bed plotters use either one rectangular sheet of paper or a fanfold stack of paper for plotting. The flat-bed plotter that uses one sheet generates lines on the x-axis by moving the entire carriage rod. On the flat-bed plotter of the fanfold type, lines along the x-axis are generated by moving the paper past the pen carriage.

Both types generate lines along the y-axis by moving the pen carriage along the carriage rod. Drum plotters use a roll of paper which unrolls from a supply reel and rolls onto a take-up reel where it may be removed. Movement along the x-axis is accomplished by moving the paper past the pen carriage, along the drum. Y-axis movement is accomplished by moving the pen carriage along the carriage rod.

INSTRUCTIONS

The plotter is driven by a controller which contains a 6-bit Command Buffer, two bits for each axis of movement.

One I/O instruction is used in programming the plotter. This instruction loads the controller's Command Buffer with the desired pen movement command.

The plotter controller's Busy and Done flags are controlled by the device flag commands as follows:

f=S Set the Busy flag to 1, the Done flag to 0, and initiate the pen movement command contained in the Command Buffer.

f=C Set both the Busy and Done flags to 0 without affecting the contents of the Command Buffer.

f=P No effect.

LOAD COMMAND BUFFER

DOA<f> ac, PLT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	0	F		0	0	1	1	0	1

Bits 10-15 of the specified AC are loaded into the controller's Command Buffer. Bits 0-9 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										RAISE PEN	LOWER PEN	-Y	+Y	-X	+X

Bits	Name	Command If Set To 1
0-9	---	Reserved for future use.
10	Raise Pen	Raise the pen off the paper.
11	Lower Pen	Lower the pen to the paper.
12	-Y	Move 1 step in the minus Y direction.
13	+Y	Move 1 step in the plus Y direction.
14	-X	Move 1 step in the minus X direction.
15	+X	Move 1 step in the plus X direction.

PROGRAMMING

When the plotter has been placed on-line and the operator has positioned the paper, the pen and/or paper may be moved by the combination of a LOAD COMMAND BUFFER instruction (DOA) and a Start command. A LOAD COMMAND BUFFER instruction will load bits 10-15 of the specified accumulator into the controller's Command Buffer, ignoring the other bits. Once loaded, a Start command may be issued to move one step in the direction specified by the contents of the Command Buffer. A Start command sets the Busy flag to 1, the Done flag to 0, and sends the command to the plotter. When the plotter has completed the command, the Busy flag is set to 0, and the Done flag is set to 1, thus initiating a program interrupt request. Since a Start command does not alter the contents of the Command Buffer, a series of Start commands may be issued to repeat any step.

TIMING

The time required for a plotter to execute a command depends upon the plotting speed of the particular plotter being used. The following time intervals are measured from the time the Start command is issued to the time the Done flag is set to 1. If the plotting speed is 200 steps/second, the time required to execute any x-axis, y-axis or combination of x- and y-axis movement is 5ms. If the plotting speed is 300 steps/second, this time interval is 3.3ms. All pen-raising and pen-lowering commands in all plotters require 100ms for execution.

CONSIDERATIONS

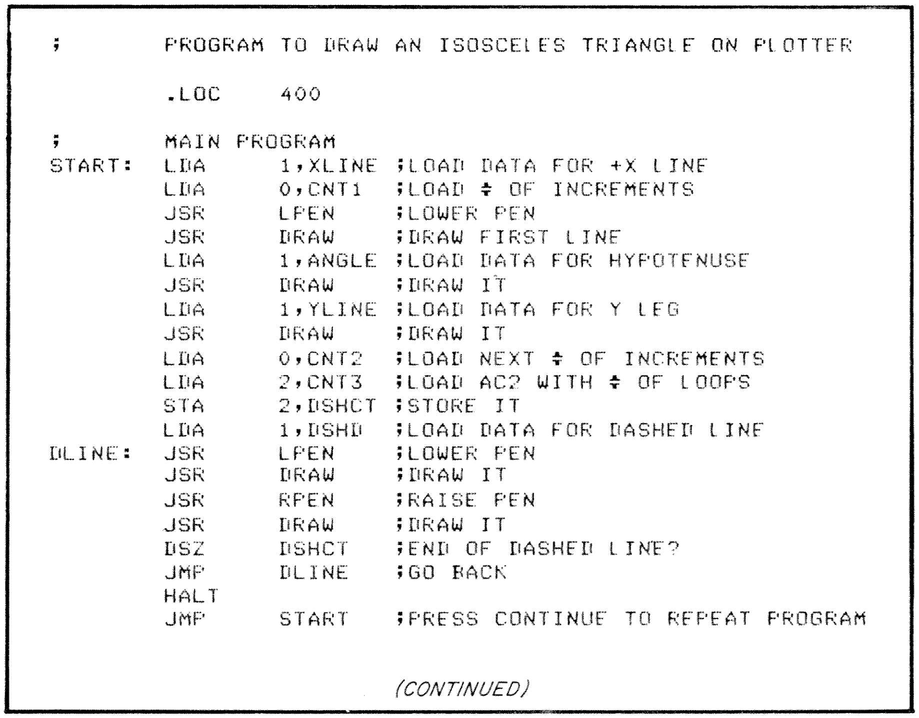
Before the plotter is placed on-line, the operator should position the pen at a starting position which is determined by the greatest negative value of both x and y used by the plotting routine. An allowance should be made to provide for margins. After the plot is completed, the pen should always be raised to prevent damaging the plot, either by blots on the paper, or when the plot is removed.

Dotted lines may be generated by issuing a pen-raising or pen-lowering command with either an x-axis or a y-axis movement. The result of this type of command will be a line segment which is shorter than the basic step size. If a dotted line or if a line segment which is less than the step size is not wanted, the program should avoid issuing such a command.

If the program loads the Command Buffer with contradictory commands such as a plus x and a minus x together, the pen will not move along that axis. However, if a third command is specified without a contradiction, the plotter will execute that third command.

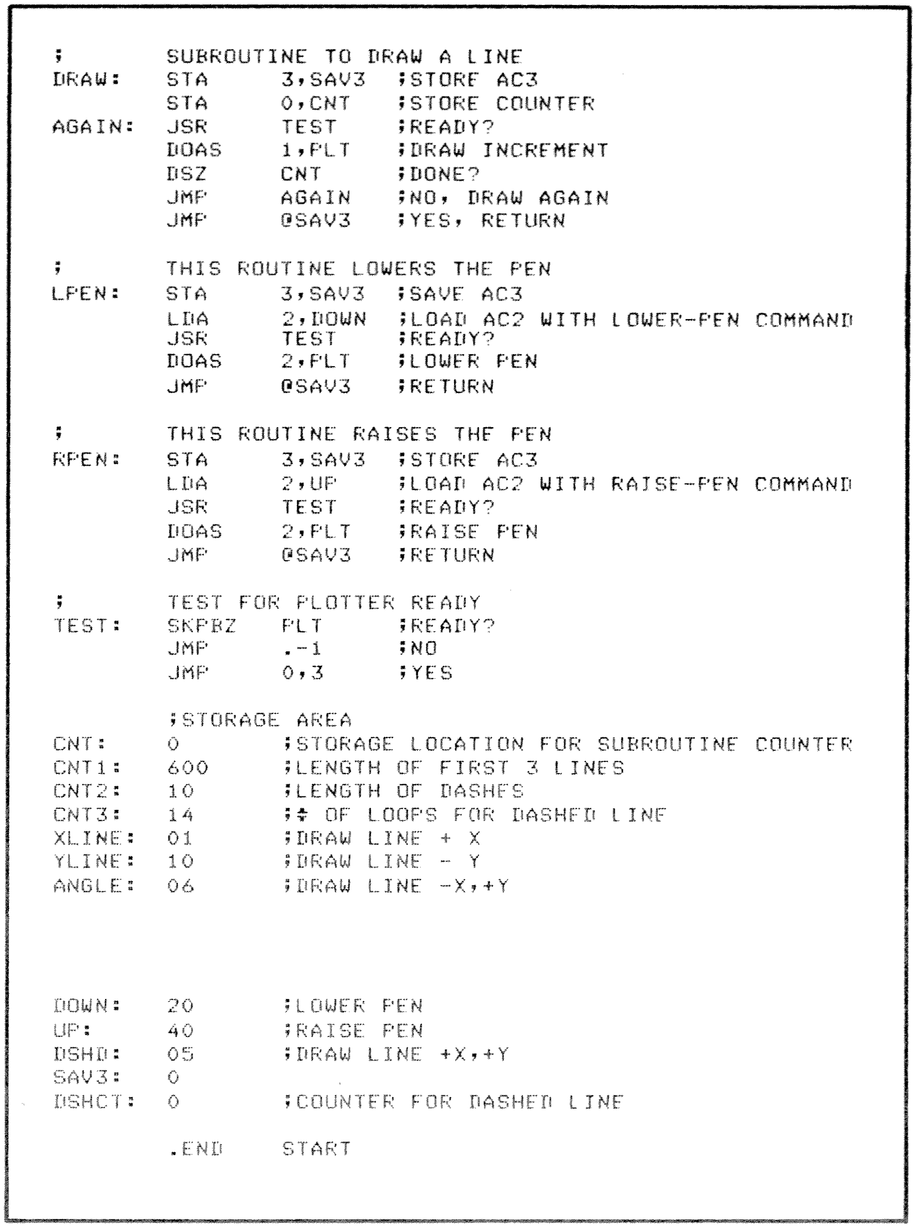
PROGRAMMING EXAMPLE

The following program illustrates the use of the incremental plotter as a stand-alone operation. The routine draws an isosceles right triangle with a dotted line dividing it in two.



```
;        PROGRAM TO DRAW AN ISOSCELES TRIANGLE ON PLOTTER

         .LOC     400

;        MAIN PROGRAM
START:   LDA      1,XLINE ;LOAD DATA FOR +X LINE
         LDA      0,CNT1  ;LOAD ÷ OF INCREMENTS
         JSR      LPEN    ;LOWER PEN
         JSR      DRAW    ;DRAW FIRST LINE
         LDA      1,ANGLE ;LOAD DATA FOR HYPOTENUSE
         JSR      DRAW    ;DRAW IT
         LDA      1,YLINE ;LOAD DATA FOR Y LEG
         JSR      DRAW    ;DRAW IT
         LDA      0,CNT2  ;LOAD NEXT ÷ OF INCREMENTS
         LDA      2,CNT3  ;LOAD AC2 WITH ÷ OF LOOPS
         STA      2,DSHCT ;STORE IT
         LDA      1,DSHD  ;LOAD DATA FOR DASHED LINE
DLINE:   JSR      LPEN    ;LOWER PEN
         JSR      DRAW    ;DRAW IT
         JSR      RPEN    ;RAISE PEN
         JSR      DRAW    ;DRAW IT
         DSZ      DSHCT   ;END OF DASHED LINE?
         JMP      DLINE   ;GO BACK
         HALT
         JMP      START   ;PRESS CONTINUE TO REPEAT PROGRAM
```

(CONTINUED)



```
;       SUBROUTINE TO DRAW A LINE
DRAW:   STA     3,SAV3  ;STORE AC3
        STA     0,CNT   ;STORE COUNTER
AGAIN:  JSR     TEST    ;READY?
        DOAS    1,PLT   ;DRAW INCREMENT
        DSZ     CNT     ;DONE?
        JMP     AGAIN   ;NO, DRAW AGAIN
        JMP     @SAV3   ;YES, RETURN

;       THIS ROUTINE LOWERS THE PEN
LPEN:   STA     3,SAV3  ;SAVE AC3
        LDA     2,DOWN  ;LOAD AC2 WITH LOWER-PEN COMMAND
        JSR     TEST    ;READY?
        DOAS    2,PLT   ;LOWER PEN
        JMP     @SAV3   ;RETURN

;       THIS ROUTINE RAISES THE PEN
RPEN:   STA     3,SAV3  ;STORE AC3
        LDA     2,UP    ;LOAD AC2 WITH RAISE-PEN COMMAND
        JSR     TEST    ;READY?
        DOAS    2,PLT   ;RAISE PEN
        JMP     @SAV3   ;RETURN

;       TEST FOR PLOTTER READY
TEST:   SKPBZ   PLT     ;READY?
        JMP     .-1     ;NO
        JMP     0,3     ;YES

        ;STORAGE AREA
CNT:    0       ;STORAGE LOCATION FOR SUBROUTINE COUNTER
CNT1:   600     ;LENGTH OF FIRST 3 LINES
CNT2:   10      ;LENGTH OF DASHES
CNT3:   14      ;÷ OF LOOPS FOR DASHED LINE
XLINE:  01      ;DRAW LINE + X
YLINE:  10      ;DRAW LINE - Y
ANGLE:  06      ;DRAW LINE -X,+Y



DOWN:   20      ;LOWER PEN
UP:     40      ;RAISE PEN
DSHD:   05      ;DRAW LINE +X,+Y
SAV3:   0
DSHCT:  0       ;COUNTER FOR DASHED LINE

        .END    START
```

SECTION IV

MAGNETIC TAPE

- INDUSTRY COMPATIBLE MAGNETIC TAPE SUBSYSTEMS
- DGC CASSETTE SUBSYSTEM

This page intentionally left blank

INTRODUCTION TO MAGNETIC TAPES

Data General offers two very different types of magnetic tape subsystems, called "industry compatible" and "cassette". Since the two types are different, they are treated separately here.

INDUSTRY COMPATIBLE MAGNETIC TAPE TRANSPORTS

Industry compatible transports are popular for storing large quantities of information which the system can afford to have serially accessible. They are called industry compatible because they conform to an industry standard established so that such tapes can be moved among systems with compatible drives. These drives are therefore an invaluable tool for passing information among normally incompatible computer systems.

The basic recording medium is a magnetic material coated on one side of a long 1/2 inch strip of mylar (usually) tape. The tape is held on large interchangeable reels (up to 2400 feet per reel) which are mounted on the supply hub of any conforming transport. When the transport is recording or reading information, the tape is moved from the supply reel past read/write heads to a take-up reel. As it moves, the heads define parallel data tracks along the surface of the tape. There are either seven or nine tracks on the tape, each track having both a read head and a write head.

Two systems are used for recording data on tape: Non Return to Zero for ones (NRZI) and Phase Encoded (PE). These systems format the tape and record data bits differently. A tape written on a transport using one system can not be read on a transport using the other system. In general, the methods used for PE recording allow greater data density on the tape (higher bpi) than NRZI so that a tape can store more information.

Data Formats

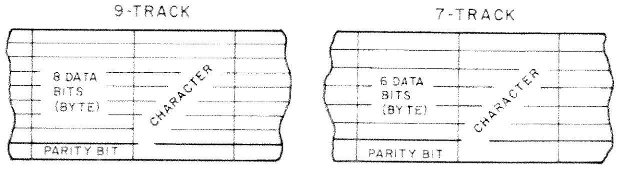
Bits

Data is stored as a "magnetic event" on the tape by the write head in the transport. As the tape moves past the write head, a sequence of data bits are written along the length of the tape. The number of data bits per inch (bpi) determines the data density for that transport. Some transports can be selected to operate at two densities.

Industry compatible tape transports contain either 7 or 9 write heads, allowing simultaneous recording of a number of parallel tracks along the length of the tape. The data bits written simultaneously by a number of heads, one bit in each track, define a character on the tape. Each character, therefore, appears across the width of the tape.

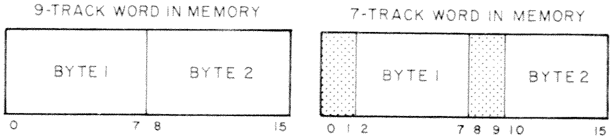
Characters and Bytes

A character is generally composed of a number of data bits and one parity bit. The data bits in a character are collectively called a byte. Seven track magnetic tape contains a 6-bit byte of data and a parity bit in each character; a 9-track tape contains an 8-bit byte of data and a parity bit.



DG-00975

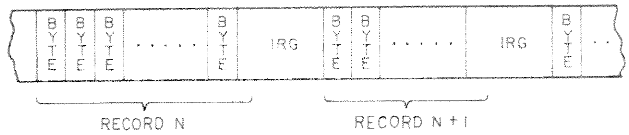
Since DGC computers utilize a 16-bit word length, two bytes from the tape are used to form one computer word as shown below. When a 9-track tape is used the two 8-bit bytes fill one computer word. However, 7-track tapes provide only 12 data bits for each computer word. The remaining four bits are ignored when writing data on tape. This means that all the 16-bit computer words have to be reformatted when writing a 7-track tape to avoid losing the contents of those four bits.



DG-00976

Records

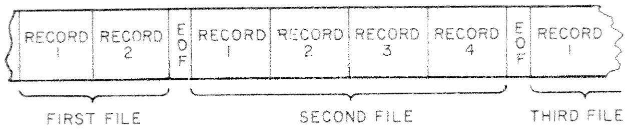
Because the amount of data stored on magnetic tape usually contains several related computer words, the pairs of bytes on the tape for each word are grouped together to form records. A record consists of a number of words (pairs of bytes) which are separated from other records by gaps on the tape. The tape transport can only stop the tape in one of these inter-record gaps (IRG). The record is the smallest possible unit of information addressable on the tape.



DG-00977

Files

Records can be grouped together in files. A file consists of related records and is separated from other files by delimiters called End Of File (EOF) marks.



DG-00978

Beginning and End Of Tape Markers

The primary reference point on a magnetic tape is the Beginning Of Tape (BOT) marker. This indicator is a reflective tab, placed along one edge of the tape, which provides an absolute reference point for all tape operations. When an operator loads a tape on a transport, the transport positions the tape at the BOT mark. Since this placement occurs automatically on loading, the BOT position is sometimes referred to as the Load Point.

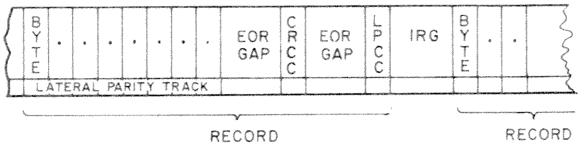
A similar reflective marker is used near the end of a tape. This indicator, called End Of Tape (EOT), is placed along the opposite edge of the tape from the BOT marker. The function of the EOT marker is to provide a warning to the programmer that the tape should not be moved forward lest it come off the supply reel.

Error Checking

NRZI

The tape passes the read heads immediately after it passes the write heads. This arrangement allows a read-after-write system of error checking which operates as follows: for NRZI systems, as each byte is received from the computer, the controller adds the correct parity bit and the resulting character is written on the tape. The character is then read by the read heads and a parity check is performed by the controller. If the parity is incorrect, an error flag is set to 1. In addition to this lateral parity checking, a parity bit is written for each track on the tape when all the characters in the record have been written. This longitudinal parity check character (LPCC), which is separated from the data by an End Of Record (EOR) gap, is also checked by the controller in the read-after-write check system. This combination of lateral and longitudinal parity checks is also performed every time a record is read.

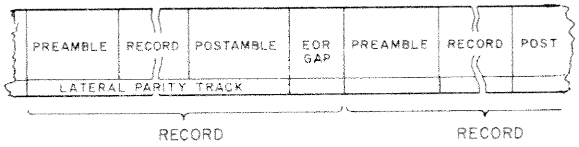
Nine-track tapes also have a cyclic redundancy check character (CRCC) written after the record to provide industry compatible tape formats.



DG-00979

PE

Phase Encoded transports perform a similar read-after-write error check system. If a character in the record contains any errors, as determined by the lateral parity, an error flag is set to 1. However when a record is read, the PE transports have the capability of both detecting and correcting a bad track in the record. This error correction system operates as follows: when one of the tracks on the tape stops supplying data bits, that track is ignored for the rest of the record and the data bits for that track are reconstructed from the parity information. Thus, the failure of a single track on the tape does not mean a loss of data. If more than one track fails in a record, the data is lost. The preamble and postamble surrounding a record are used to synchronize the error correction system. They are transparent to the programmer.



DG-00980

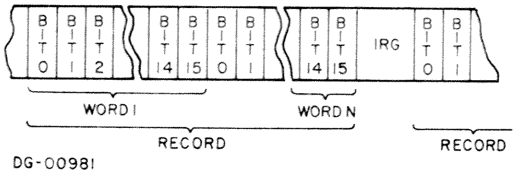
DGC CASSETTE TAPE TRANSPORTS

The DGC cassette tape subsystem is offered for applications where large amounts of data storage are not needed but a paper tape subsystem appears unwieldy. The data is stored on a single track of magnetic tape which is contained in a removable cassette. Each cassette contains the supply reel of tape as well as the take-up reel. When the transport is recording or reading information, the tape is moved from the supply reel past read/write heads, to a take-up reel. As it moves, the heads define a single data track along its length. There is only one track which has a separate read and write head.

Data Formats

Bits and Words

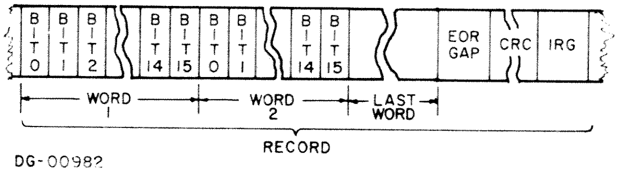
Since a DGC cassette transport records data on a single track along a length of magnetic tape, each bit of a 16-bit word is written sequentially along that track. Words are written consecutively with no gaps between them.



DG-00981

Records

Related words, forming a record, are grouped together and each record is separated from the adjacent records by gaps on the tape called inter-record (IRG) gaps. Just as in the industry compatible magnetic tapes, the record is the smallest addressable unit of information on a DGC cassette.



DG-00982

Files

Records can be grouped together in files. A file usually consists of related records and is separated from other files by delimiters called End Of File (EOF) marks.

Beginning and End Of Tape Markers

Just as the larger industry compatible tapes have BOT and EOT markers, each cassette tape has a reflective leader and trailer which delimit the end of its tape. Since the cassette can be removed from the transport when the tape is positioned in any inter-record gap, each time a cassette is placed on a transport, the programmer should ascertain that the tape is at the BOT position before attempting any operations.

Error Checking

The read head is positioned so that the tape passes under it just after it passes the write head. This allows a read-after-write error check system which operates as follows: as the individual bits are written on the tape, the controller uses the bits to calculate a cyclic redundancy checkword (CRC). Once all the words in a record have been written, and an End Of Record (EOR) gap is inserted, this checkword is written.

While writing the record, the read portion of the transport reads the data fram the tape and the controller independently calculates another checkword and compares it to the original checkword read from the tape at the end of the record. If these two checkwords differ, an error flag is set to 1. A similar check is performed each time a record is read.

Summary of Magnetic Tape Subsystem Specifications

Subsystem	Controller	Adapter	Transport	Number of Tracks	Density (bpi)	Tape Speed (ips)	Transfer Rate (Words/Sec)		Maximum Number of Transports/ Subsystem	Storage Capacity/Unit (Million Words)	
							556 bpi	800 bpi		556 bpi	800 bpi
NRZI	4030	PROVIDED	4030I/P	7	556 or 800	45	12,600	18,000	8	7.5	11
			4030J/R	9	800	45	N/A	18,000		N/A	11
			4030K/S	7	556 or 800	12.5	3,480	5,000		7.5	11
			4030L/T	9	800	12.5	N/A	5,000		N/A	11
			4030M/U	7	556 or 800	75	21,000	30,000		7.5	11
			4030N/W	9	800	75	N/A	30,000		N/A	11
			6020/6022 6024	7	556 or 800	75	21,000	30,000		7.5	11
			6021/6023 6025	9	556 or 800	75	21,000	30,000		7.5	11
PE	4196	Provided with 4196A only	4196A/B	9	1600	45	36,000 @ 1600bpi		4	22 @ 1600bpi	
Cassette	4076	Provided	4080, 4081 4084, 4087 4088, 4089	1	430 (Average)	30 (Average)	750		8	.067	

DG-00984

This page intentionally left blank

INDUSTRY COMPATIBLE MAGNETIC TAPE SUBSYSTEMS

SUMMARY

MNEMONIC (FIRST CONTROLLER)	MTA
DEVICE CODE (FIRST CONTROLLER)	22_8
MNEMONIC (SECOND CONTROLLER)	MTA1
DEVICE CODE (SECOND CONTROLLER)	62_8
PRIORITY MASK BIT	10
MAXIMUM REEL SIZE (INCHES)	10 1/2
WORDS/RECORD	2 to 65,536
MAXIMUM STORAGE CAPACITY (WORDS)	7,500,000 to 22,000,000
MAXIMUM TRANSFER RATE (WORDS/SEC)	3,480 to 36,000
MAXIMUM ALLOWABLE DATA CHANNEL LATENCY (μSEC)	432 to 13.8

DATA FORMATS

SEVEN TRACK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨		BYTE 1						▨		BYTE 2					

NINE TRACK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BYTE 1								BYTE 2							

ACCUMULATOR FORMATS

SPECIFY COMMAND AND UNIT .... (DOA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨									PARITY NRZ1	COMMAND			UNIT		

COMMANDS

000	Read	100	Space Reverse
001	Rewind	101	Write
010	Reserved	110	Write EOF
011	Space Forward	111	Erase

READ STATUS .... (DIA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ERROR	DATA LATE	REWIND-ING	ILL-EGAL	HIGH DEN-SITY	PARITY ERROR	END OF TAPE	END OF FILE	BEGIN OF TAPE	9 TRACK	BAD TAPE	SEND CLOCK	FIRST CHAR.	WRITE LOCK	ODD CHAR	UNIT READY

LOAD MEMORY ADDRESS COUNTER .... (DOB)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAINT	MEMORY ADDRESS														

READ MEMORY ADDRESS COUNTER ..... (DIB)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MEMORY ADDRESS															

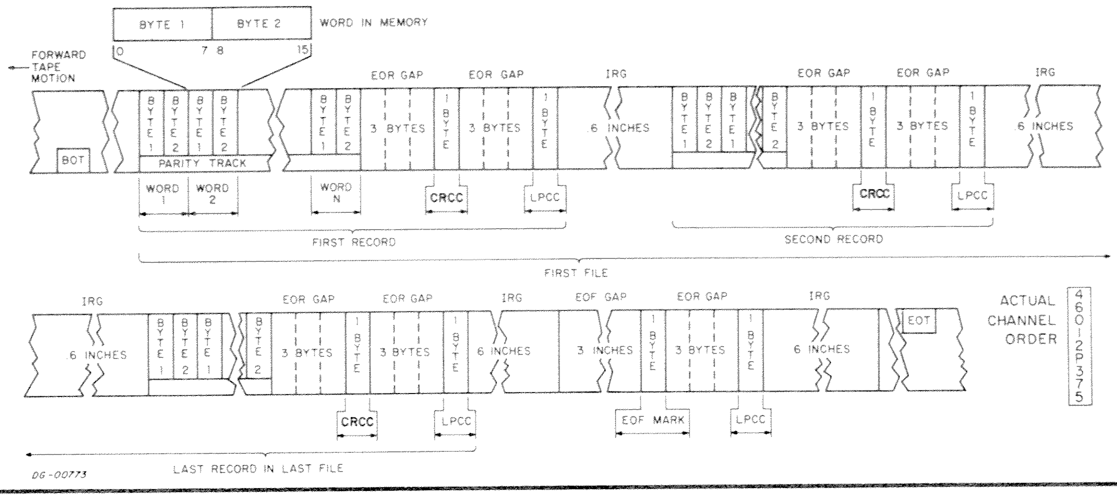
LOAD WORD COUNTER .... (DOC)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OPTIONAL FOR PE TAPES				– WORD COUNT											

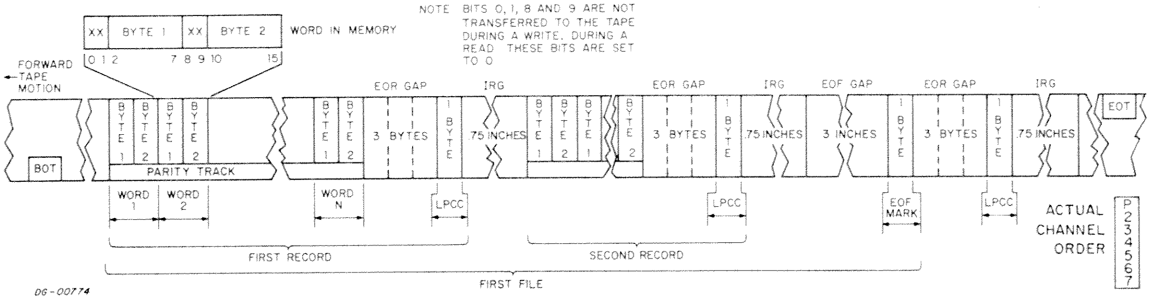
S, C AND P FUNCTIONS

S	Set Busy to 1, Done to 0 and start the command.
C	Set all error flags to 0, select unit 0 as the addressed transport and Read command.
P	No effect.

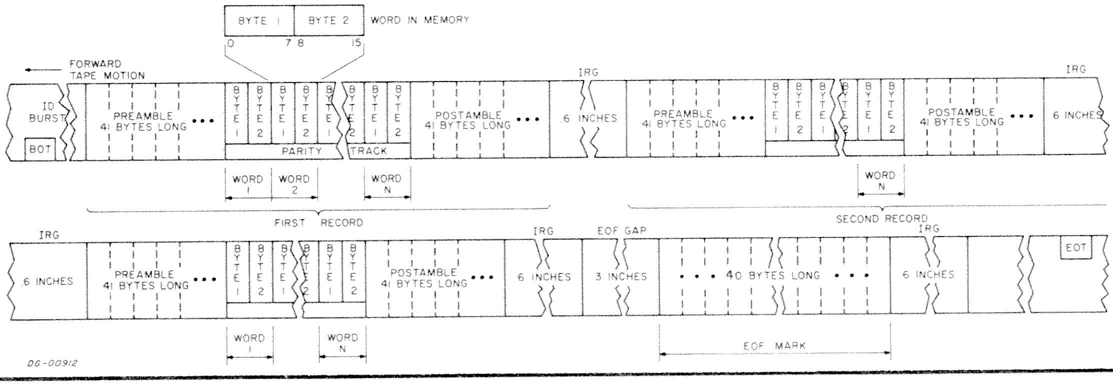
NINE TRACK NRZI TAPE FORMAT



SEVEN TRACK NRZI TAPE FORMAT



NINE TRACK PE TAPE FORMAT



Abbreviation	Name	Description
BOT	BEGINNING OF TAPE	A reflective marker at the beginning of the tape.
CRC	CYCLIC REDUNDANCY CHECK CHARACTER	A checksum calculated from and used to validate all the data in the preceding record on 9-track tapes.
EOR	END OF RECORD GAP	A blank section of tape written by the controller to indicate either the end of the data in a record or the end of the CRC.
EOT	END OF TAPE	A reflective marker at the end of the tape.
ID BURST	IDENTIFICATION BURST	A code written by the controller on phase encoded tapes to identify them as such.
IRG	INTER-RECORD GAP	A blank section of tape written by the controller after the postamble when the tape transport stops and restarts for the next record.
	PREAMBLE	A forty-one byte field used on phase encoded tapes to synchronize the controller. It is written by the controller before each record.
	POSTAMBLE	A forty-one byte field used in phase encoded tapes to signify the end of the record. It is written by the controller.
LPCC	LONGITUDINAL PARITY CHECK CHARACTER	A character which provides the correct longitudinal parity for each track on the tape.

INTRODUCTION

Data General supplies two types of industry compatible tape subsystems, NRZI and PE. NRZI tape drives are available in both 7- and 9-track format; at 556 and 800 bits per inch (bpi); and in even or odd parity. The PE tape drive operates at 1600bpi in 9-track format. Depending on the particular transport, the data transfer rate ranges from 3,480 words/second to 36,000 words/second.

The controller used in the NRZI subsystem can simultaneously accommodate up to eight transports in any combination. The controller used in the PE subsystem can accommodate up to 4 transports. In either type of subsystem, only one transport can be reading, writing or spacing at any one time, but any number of transports can be rewinding simultaneously.

Records on the tape are composed of groups of words ranging in length from 2 to 65,536 words per record, depending on the subsystem. Files are composed of groups of records. The format of the record and file structure is shown on the opposite page. The number of files which can be placed on a reel of tape is dependent on the length of the tape, the information density, the number of words per record and the number of records per file. A full 10 1/2 inch reel of 1.5 mil tape can store more than 11 million 16-bit words in an NRZI subsystem or more than 22 million 16-bit word in a PE subsystem.

Data is verified during Write operations by a combination of lateral and longitudinal parity checks in a read-after-write error checking system. The same system is used when the tape is read.

INSTRUCTIONS

The tape transport controller contains four registers: a 15-bit Memory Address Counter, a 16-bit Status Register, a 12- or 16-bit Word Counter, and a 6- or 7-bit combined Command/Transport Select Register. The Memory Address Counter is self-incrementing and contains the memory location of the next word to be either read from or written on the tape. The Status Register contains all the information flags for the controller and the selected transport. The Word Counter contains the two's complement of the number of words to be read from or written on the tape or the two's complement of the number of records to be skipped in a spacing operation. The combined Command/Transport Select Register contains the last command issued to the tape subsystem and the unit number of the transport currently selected.

Five instructions are used to program data channel transfers to and from the tape subsystem. Three of these instructions are used to supply all of the necessary data to the controller for any tape operation. The remaining two instructions allow the program to determine, in detail, the current state of the selected tape transport.

The tape subsystem controller's Busy and Done flags are controlled using two of the device flag commands as follows:

f=S Set the Busy flag to 1, the Done flag to 0 and set the transport in operation for the command contained in the Command Register. Providing the Illegal flag is set to 0, all error indicating flags in the Status Register are set to 0.

f=C Set the Busy flag, the Done flag, and all error indicating flags in the Status Register to 0. The error indicating flags are: Data Late, Illegal, Parity Error, Bad Tape, End Of File, and Odd Character. After a Clear command is issued, the selected transport is unit 0 and the specified command is Read.

f=P No effect.

SPECIFY COMMAND AND UNIT

DOA <f> ac, MTA

0	1	1	AC		0	1	0	F		0	1	0	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 9-15 of the specified AC are loaded into the combined Command/Transport Select Register. Bits 0-8 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

									PARITY NRZI	COMMAND			UNIT		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-8	----	Reserved for future use.
9	Parity (NRZI)	For NRZI, 0 selects odd parity; 1 selects even parity. If even parity is selected, a series of null bytes in a record could be interpreted as an inter-record gap, resulting in an error condition. This bit is ignored by the Phase Encoded controller.
10-12	Command	Select the command for the selected transport as follows: 000 Read 001 Rewind 010 Reserved for future use 011 Space forward 100 Space reverse 101 Write 110 Write End Of File 111 Erase
13-15	Unit	Select transport 0-7 for NRZI or 0-3 for PE.

READ STATUS

DIA <f> ac, MTA

0	1	1	AC		0	0	1	F		0	1	0	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Status Register are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

ERROR	DATA LATE	REWIND-ING	ILL-EGAL	HIGH DEN-SITY	PARITY ERROR	END OF TAPE	END OF FILE	BEGIN OF TAPE	9 TRACK	BAD TAPE	SEND CLOCK	FIRST CHAR.	WRITE LOCK	ODD CHAR	UNIT READY
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Meaning When 1
0	Error	One or more of the bits 1, 3, 5, 6, 7, 8 and 10 is 1.
1	Data Late	The data channel failed to respond in time to a data channel request.
2	Rewinding	The selected transport is currently rewinding.
3	Illegal	A Start command was issued to the selected transport when one of the following conditions existed: 1. The transport was not ready. 2. The command was Space Reverse and the tape was at the beginning of tape leader. 3. The command was Write, Write End Of File or Erase when the tape was write-protected.
4	High Density	The selected NRZI transport is set to Read or Write at 800bpi. This bit is always 1 for PE transports (1600bpi).
5	Parity Error	One or more bytes in the record did not have the correct parity, or in NRZI, tapes the LPCC read from the tape after the record did not match the character calculated by the controller.
6	End Of Tape	The transport has reached or passed the End Of Tape mark. Executing either a Space Reverse or a Rewind operation will set this bit to 0.
7	End Of File	The transport has encountered an End Of File mark in reading, spacing or after writing an EOF mark.
8	Begin Of Tape	The tape is at the beginning of tape mark.
9	9-Track	The selected transport is set to Read or Write a 9-Track tape. This bit is always 1 for PE transports.
10	Bad Tape	The section of tape just processed is of poor quality.
11	Send Clock	Used for maintenance.
12	First Character	Used for maintenance.
13	Write Lock	The tape on the selected transport is write-protected.
14	Odd Character	The record just read or written contains an odd number of bytes.
15	Unit Ready	The selected transport is ready.

LOAD MEMORY ADDRESS COUNTER

DOB<f> ac,MTA

0	1	1	AC		1	0	0	F		0	1	0	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-15 of the specified AC are loaded into the Memory Address Counter. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

MAINT	MEMORY ADDRESS														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0	Maintenance	When set to 1, this bit places the NRZI subsystem in the maintenance mode of operation. This mode allows the cyclic redundancy check character used on 9-track NRZI tapes to be read into the memory location following the last data word in memory for the record.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer

LOAD WORD COUNTER

DOC<f> ac,MTA

0	1	1	AC		1	1	0	F		0	1	0	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 4-15 of the specified AC are loaded into the controller's Word Counter. Bits 0-3 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The specified AC must contain the two's complement of the number of words to be transferred with the stipulation that the minimum number of words to be transferred is 2 and the maximum is 4096 for NRZI and 65,536 for PE.

During a spacing operation, the Word Counter acts as a record counter. The specified AC must contain the two's complement of the number of records to be skipped with the stipulation that the minimum number of records to be skipped is 1 and the maximum is 4096 for NRZI and 65,536 for PE. The format of the specified AC is as follows:

OPTIONAL FOR PE TAPES				— WORD COUNT											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-3	----	Reserved for future use in NRZI controller.
4-15 or 0-15	-Word Count	Two's complement of number of words to be transferred or records to be skipped.

READ MEMORY ADDRESS COUNTER

DIB<f> ac,MTA

0	1	1	AC		0	1	1	F		0	1	0	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Memory Address Counter are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. When the Memory Address Counter is read after a Read or a Write operation, the contents point to a memory location one greater than the location of the last word transferred. The format of the specified AC is as follows:

MEMORY ADDRESS															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

PROGRAMMING

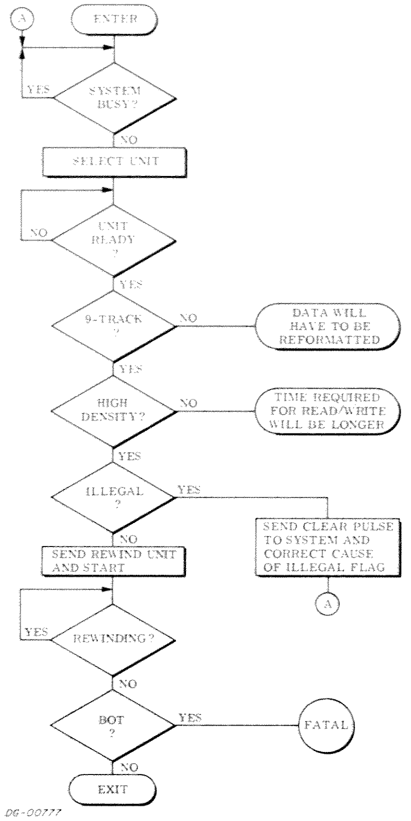
The preparation of a magnetic tape subsystem for a data channel transfer can be divided into three distinct phases: I, initializing the transport, II, positioning the tape; and III, specifying the length of the transfer, the starting memory address, and issuing the tape command. Once the first two phases are completed, the program can repeatedly execute the third phase. The results of issuing commands in each phase should be checked for errors before proceeding to the next phase.

Phase I: Initializing

The desired transport is selected in Phase I by issuing a SPECIFY COMMAND AND UNIT instruction (DOA) to the controller. The Status Register is examined by a READ STATUS instruction (DIA) to determine if the tape is 7- or 9-track, high (800bpi) or low (556bpi) density, and whether the transport is in the ready state. The 7- or 9-track test is used to determine if the data has to be reformatted; the high or low density test is used to determine the times needed for data transfer; and the ready test determines if the transport is available to the program. The transport should be initialized by issuing a Rewind command if the tape is not located at the Beginning Of Tape mark. The Rewind command can be executed providing the transport is ready and the Illegal flag is set to 0. If the Illegal flag is set to 1 in the Status Register, a Clear command can be issued. The transport is then selected and a Rewind command specified by issuing a SPECIFY COMMAND AND UNIT instruction. The operation is initiated by issuing a Start command. The Start command sets the Rewinding flag to 1 in the Status Register but does not set either the Busy flag or the Done flag to 1. The controller does not initiate a program interrupt request when the rewinding operation is completed. However, the Rewinding flag will be set to 0 in the Status Register upon completion of the program.

Once the tape has been rewound and no error conditions are detected in the Status Register, the program may proceed to Phase II.

INITIALIZE


A
ENTER
SYSTEM BUSY?
YES
NO
SELECT UNIT
UNIT READY ?
NO
YES
9-TRACK ?
NO
DATA WILL HAVE TO BE REFORMATTED
YES
HIGH DENSITY?
NO
TIME REQUIRED FOR READ/WRITE WILL BE LONGER
YES
ILLEGAL ?
YES
SEND CLEAR PULSE TO SYSTEM AND CORRECT CAUSE OF ILLEGAL FLAG
A
NO
SEND REWIND UNIT AND START
YES
REWINDING ?
NO
BOT ?
YES
FATAL
NO
EXIT
DG-00777

Phase II: Position the Tape

In order to access the desired record on the magnetic tape for a Read or a Write operation, the tape should be spaced to the position immediately preceding that record. During the spacing operation, the Word Counter is used as a record counter, incrementing once for each record skipped. The spacing operations terminates when the Word Counter overflows or when an End Of File mark is encountered. The minimum number of records which can be skipped in one operation is one and the maximum is 4096 for NRZI or 65,536 for PE.

The spacing operation is performed as follows: the command is loaded into the Command Register with a SPECIFY COMMAND AND UNIT instruction (DOA), and the two's complement of the number of records to be skipped is specified with a LOAD WORD COUNTER instruction (DOC). The spacing operation is initiated with a Start command appended to the last of these instructions issued. The Busy flag is set to 1 and the Done flag is set to 0 during spacing. Upon completion of the operation, the Busy flag is set to 0 and the Done flag is set to 1, thus initiating a program interrupt request.

If the desired record is contained in another file, the Space Forward command must be issued at least once for each file to be skipped. When a file is to be skipped, the Word Counter must be loaded with the two's complement of a count greater than or equal to the number of records contained in that file. Since a file may contain more than the maximum number of records which can be skipped in one operation, the Space Forward command may have to be executed several times (reloading the Word Counter each time) before the file is skipped. Every time an End Of File mark is encountered in a spacing operation, the operation stops, both the End Of File flag and the Error flag are set to 1 in the Status Register, the Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

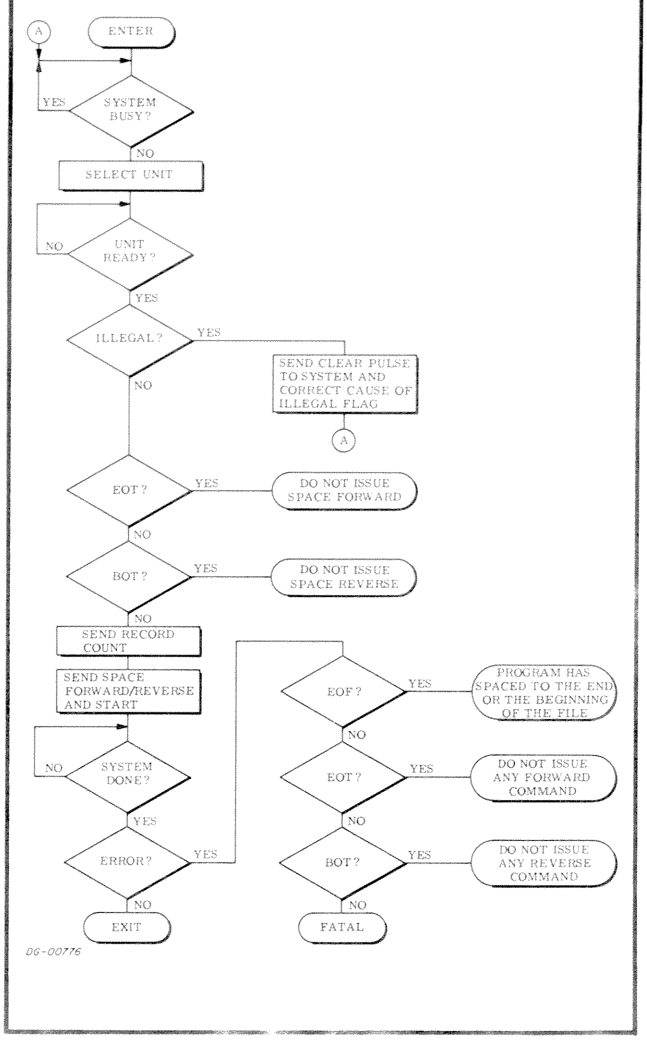
Once the tape drive has reached the correct file, the tape should be positioned immediately before the desired record in that file.

The Space Reverse command operates in a manner entirely analogous to the Space Forward command. After a Space Reverse command is executed as many times as is necessary to skip a file, the tape is positioned immediately after the last record in the preceding file.

Care should be taken so that a Space Forward command is not executed when the End of Tape flag is set to 1 in the Status Register.

Once the tape is in position for a Read or a Write operation and no errors are detected in the Status Register, the program may proceed to Phase III.

SPACE FORWARD/REVERSE


A
ENTER
YES
SYSTEM BUSY?
NO
SELECT UNIT
NO
UNIT READY?
YES
ILLEGAL?
YES
SEND CLEAR PULSE TO SYSTEM AND CORRECT CAUSE OF ILLEGAL FLAG
A
NO
EOT?
YES
DO NOT ISSUE SPACE FORWARD
NO
BOT?
YES
DO NOT ISSUE SPACE REVERSE
NO
SEND RECORD COUNT
SEND SPACE FORWARD/REVERSE AND START
NO
SYSTEM DONE?
YES
ERROR?
YES
NO
EXIT
EOF?
YES
PROGRAM HAS SPACED TO THE END OR THE BEGINNING OF THE FILE
NO
EOT?
YES
DO NOT ISSUE ANY FORWARD COMMAND
NO
BOT?
YES
DO NOT ISSUE ANY REVERSE COMMAND
NO
FATAL
DG-00776

Phase III: Read or Write

Phase III consists of specifying the starting location in memory for the first word to be transferred, loading the Word Counter with the two's complement of the number of words to be transferred and issuing either a Read or a Write command together with a Start command to the desired transport.

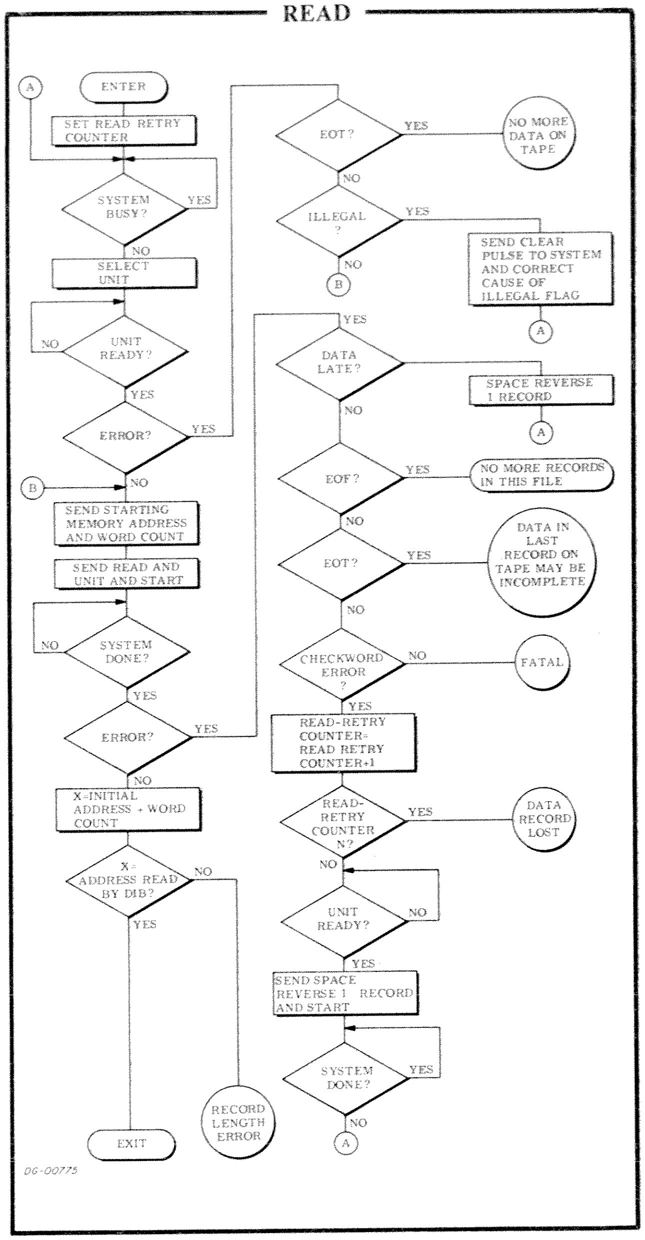
Read

Read operations are performed as follows: the storage location in memory for the first word to be read from the tape is specified by a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The two's complement of the number of words to be read is then specified by a LOAD WORD COUNTER instruction (DOC). If the record to be read is of unknown length, giving the two's complement of the longest possible record, i.e., 0, assures that the entire record will be read. The desired transport is selected and the Read command is loaded into the Command Register with a SPECIFY COMMAND AND UNIT instruction (DOA). A Start command in the last of these instructions initiates the Read operation.

Once the Read command is initiated, the tape is moved past the heads and the first two bytes in the record are assembled into one word. The word is then written into memory via the data channel. Each time a word is transferred to memory, the Word Counter is incremented by one.

The Read operation will continue until either the Word Counter overflows or the transport encounters an End Of Record gap. If the record read was of unknown length and was read by loading a zero in the Word Counter, the length can be determined by obtaining the address of the last word read into memory. This address is obtained by issuing a READ MEMORY ADDRESS COUNTER instruction (DIB). The address returned is one greater than the address of the last word written in memory by the Read operation. Therefore, subtracting the starting memory location from the address returned by a READ MEMORY ADDRESS COUNTER instruction will yield the length of the record read.

Whenever the End Of Tape flag is set to 1 in the Status Register, care should be taken to ensure that the tape does not come off the supply reel.



Write

Write operations (Write, Write End Of File, and Erase) are performed as follows: for a Write command, the storage location in memory for the first word to be written on the tape is specified by a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The two's complement of the number of words to be written is then specified by a LOAD WORD COUNTER instruction (DOC). The desired transport is selected and the Write command is loaded into the command register with a SPECIFY COMMAND AND UNIT instruction (DOA). A Start command in the last of these instructions issued initiates the Write operation.

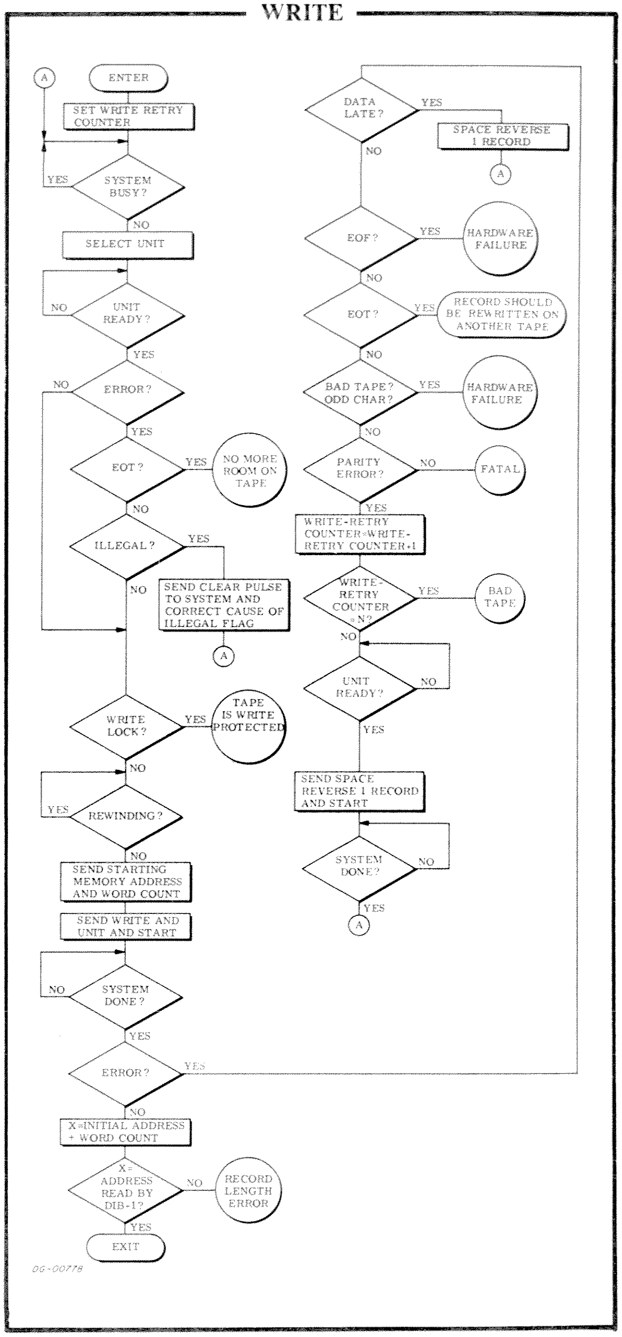
Once the Write command is initiated, a word is read from memory, the tape is moved past the heads, and the two bytes of the word are written on the tape. Each time the controller receives a word from memory, the Word Counter is incremented. Once the Word Counter overflows, the NRZI controller writes the cyclic redundancy check character and the LPCC character on 9-track tapes or, on 7-track tapes, the NRZI controller writes only the LPCC. The PE controller adds the postamble when the Word Counter overflows. In all cases, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Write End Of File and Erase commands are accomplished by selecting the desired transport and loading the desired command into the Command Register with a SPECIFY COMMAND AND UNIT instruction. A Start command initiates the operation.

The Write End Of File command erases 3 inches of tape and writes an End Of File mark on the tape after the inter-record gap. Upon completion of this operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. Since the transports perform a read-after-write error check, both the End Of File flag and the Error flag are set to 1 in the Status Register each time a correct End Of File mark is written.

NOTE An End Of File mark must be written in even parity on a 7-track NRZI tape.

The Erase command erases approximately 2 1/2 inches of tape each time it is issued. Erase can be used to skip bad sections of tape. Once the Erase operation has been completed, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.



TIMING

The timing specifications and the controller's data channel rates and latencies for the various transports are listed below. All the timing considerations for the NRZI transports are identical for both the 7- and 9-track units; the differences appear when comparing 556 and 800bpi data densities. The start delays are measured from the time a Start command is issued to the time the transport begins reading or writing a record. The stop time is divided into three segments: last character to stop, delay time, and settle time. Last character to stop is the time from the last character in the record to the beginning of the stop delay. The delay time allows the record to move completely past the read/write heads. The settle time allows the tape to stabilize in the rest position. The sum of these three times is the total stop time.

Industry Compatible Magnetic Tape Transport Specifications

Model	No. of Tracks	Tape Speed (in/sec)	Density (bpi)	Transfer Rate (Kwords/sec)		Time Word (μsec)		Maximum Data Channel Latency (μsec)		Start Time (ms)				Last Character To Stop (ms)				Stop Delay (ms)				Settle Time (ms)	Total Stop Time (ms)			
										Read		Write		Read		Write		Read		Write			Read		Write	
				556	800	556	800	556	800	556	800	556	800	556	800	556	800	556	800	556	800		556	800	556	800
4030I/P	7	45	556/800	12.6	18	80	56	120	84	22.5	15.0	15	10.0	.11	.07	3.5	3.5	2.5	1.7	5.0	3.3	13.0	15.6	14.8	21.5	19.8
4030J/R	9	45	800	---	18	---	56	---	84	---	15.0	---	10.0	---	.07	---	3.5	---	1.7	---	3.3	13.0	---	14.8	---	19.8
4030K/S	7	12.5	556/800	3.48	5	288	200	432	288	90	60	56	37	4.5	3.0	12.5	12.5	8.5	5.7	18	12	30.0	43.0	38.7	60.5	54.5
4030L/T	9	12.5	800	---	5	---	200	---	288	---	60	---	37	---	3.0	---	12.5	---	5.7	---	12	30.0	---	38.7	---	54.5
4030M/U	7	75	556/800	21	30	48	33.3	72	50	20.3	14.0	9.6	6.4	.08	.05	2.0	2.0	1.5	1.0	9.6	6.4	10.0	11.58	11.05	21.6	18.4
4030N/W	9	75	800	---	30	---	33.3	---	50	---	14.0	---	6.4	---	.05	---	2.0	---	1.0	---	6.4	10.0	---	11.05	---	18.4
6020/6022 6024	7	75	556/800	21	30	48	33.3	72	50	20.3	14.0	9.6	6.4	.08	.05	2.0	2.0	1.5	1.0	9.6	6.4	10.0	11.58	11.05	21.6	18.4
6021/6023 6025	9	75	556/800	21	30	48	33.3	72	50	20.3	14.0	9.6	6.4	.08	.05	2.0	2.0	1.5	1.0	9.6	6.4	10.0	11.58	11.05	21.6	18.4
4196A/B	9	45	1600	36		27.8		13.8		12.9		10.6		0.6		3.9		1.8		3.5		14.0	16.4		21.4	

ERROR CONDITIONS

During Initializing

If a Start command is given when the selected unit is not ready to carry out a command, both the Illegal flag and the Done flag will be set to 1. The Illegal flag can be set to 0 only with a Clear command or an I/O RESET instruction. The Clear command sets all the error indicating flags in the Status Register to 0 and sets the Command/Transport Select Register to 0. Therefore, unit 0 is the selected transport and Read is the specified command after a Clear command or an I/O RESET instruction is issued. The Status flags read by a READ STATUS instruction (DIA) pertain to unit 0 until another transport is selected.

During Positioning

Both the Illegal flag and the Error flag are set to 1 in the Status Register when a Spacing command is issued to a transport which is not ready or if a Space Reverse command is issued when the tape is at the BOT mark. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

During Reading

If the data channel does not respond in time to a data channel request, both the Data Late flag and the Error flag will be set to 1 in the controller's Status Register, but the Read operation will continue until either the Word Counter overflows or the transport encounters an End Of Record gap on the tape. The Data Late flag indicates that at least one word on the tape was not correctly transferred to memory. If the transport encounters an End Of File mark, the Error flag, together with the End of File flag, is set to 1 in the controller's Status Register. Busy is set to 0, Done is set to 1 and a program interrupt request is initiated.

If any character in the record has incorrect parity, both the Error and Parity Error flags are set to 1, and the reading of the record will continue.

Once the transport encounters the end of the data in a record or the Word Counter overflows, the NRZI controller reads the LPCC and if the LPCC does not match the LPCC calculated by the controller, the Parity Error flag is set to 1 in the Status Register. The correct LPCC for the entire record will be read even if the program reads only a portion of the record.

During Writing

If the selected transport is not ready to receive a command or, if the tape mounted on the selected transport is write-protected and a Write operation is specified, the Start command will set the Error flag, the Illegal flag and the appropriate error condition flag in the controller's Status Register to 1. Busy will be set to 0, Done will be set to 1 and a program interrupt request is initiated.

If the data channel does not respond in time to a data channel request, both the Data Late flag and the Error flag will be set to 1 in the Status Register but the Write operation will continue until the Word Counter overflows. The Data Late flag indicates that at least one word from memory was not transferred to the tape.

If the End Of File mark contains an error, the controller will interpret the mark as a short data record and a Parity Error will occur on every subsequent Read operation of that section of tape. A defective End Of File mark will also be interpreted as a data record during spacing operations. The EOF mark must be written in even parity on 7-track NRZI tapes.

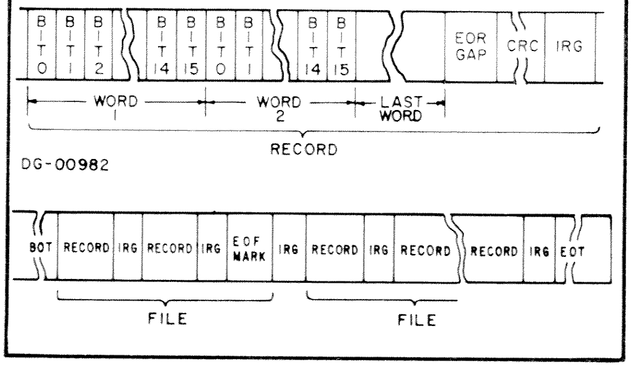
DGC CASSETTE SUBSYSTEM

INTRODUCTION

Each DGC cassette has a maximum storage capacity of 67,000 words (134,000 eight-bit bytes). The actual capacity of a DGC cassette is determined by the record and file structure used. Differences in capacity are due to the amount of tape used for inter-record gaps and End Of File marks. Reducing the number of these gaps and marks allows more data to be stored on a DGC cassette. The following table gives the capacities for several schemes of storing data on a DGC cassette.

Words/ Record	Records/ File	Files/DGC cassette	Total Words/ DGC cassette
16	16	64	16K
64	16	32	36K
128	16	22	44K
4096	14	1	56K

RECORD & FILE FORMATS



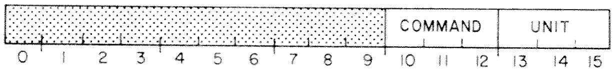
The data stored on a DGC cassette is verified by the calculation of a cyclic redundancy checkword (CRC) in a read-after-write checking system. The same system is used when a record is read.

SUMMARY

MNEMONIC (FIRST CONTROLLER)	CAS
DEVICE CODE (FIRST CONTROLLER)...	34_8
MNEMONIC (SECOND CONTROLLER) ...	CAS1
DEVICE CODE (SECOND CONTROLLER)...	74_8
PRIORITY MASK BIT	10
LENGTH OF TAPE (FEET)	200
RECORDS/FILE	1 to 4096
WORDS/RECORD	2 to 4096
MAXIMUM TRANSFER RATE (WORDS/SEC)	750
MAXIMUM STORAGE CAPACITY (WORDS/CASSETTE)	67,000
ALLOWABLE DATA CHANNEL LATENCY (μSEC)...............	28

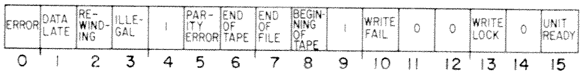
ACCUMULATOR FORMATS

SPECIFY COMMAND AND UNIT........ (DOA)

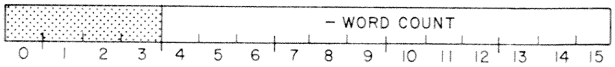


000 READ
001 REWIND
010 RESERVED
011 SPACE FORWARD
100 SPACE REVERSE
101 WRITE
110 WRITE EOF
111 ERASE

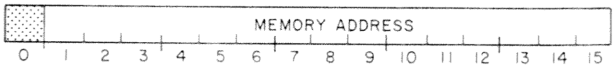
READ STATUS.......................... (DIA)



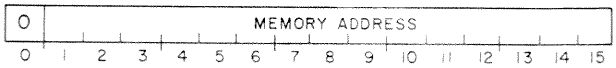
LOAD WORD COUNTER .... (DOC)



LOAD MEMORY ADDRESS COUNTER... (DOB)



READ MEMORY ADDRESS COUNTER... (DIB)



S Set Busy to 1, Done to 0 and all error indicating flags to 0 and start the command. If Illegal is set to 1, S has no effect.

C Set Busy, Done and all error indicating flags to 0, select unit 0 as the addressed transport and specify a Read command.

P No effect.

INSTRUCTIONS

The DGC cassette controller contains four registers: a 15-bit Memory Address Counter, a 16-bit Status Register, a 12-bit Word Counter, and a 6-bit combined Command/Transport Select Register. The Memory Address Counter is self-incrementing and contains the memory location of the next word to be either read from or written on the tape. The Status Register contains all the information flags for the controller and the selected transport. The Word Counter contains the two's complement of the number of words to be read from or written on the DGC cassette or the two's complement of the number of records to be skipped in a spacing operation. The combined Command/Transport Select Register contains the last command issued to the DGC cassette subsystem and the unit number of the transport currently selected.

Five instructions are used to program data channel transfers to and from the DGC cassette subsystem. Three of these instructions are used to supply all of the necessary data to the controller for any DGC cassette operation. The remaining two instructions allow the program to determine, in detail, the current state of the selected DGC cassette transport.

The DGC cassette subsystem controller's Busy and Done flags are controlled using two of the device flag commands as follows:

f=S Set the Busy flag to 1, and the Done flag to 0. If the Illegal flag is 0, all other error indicating flags are set to 0, and the transport initiates the operation specified in the Command Register. If the Illegal flag is 1, the Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

f=C Set the Busy flag, the Done flag, and all error indicating flags in the Status Register to 0. The error indicating flags are: Error, Data Late, Illegal, Checkword Error, Write Fail, and End Of File. After a Clear command is issued, the selected transport is unit 0 and the specified command is Read.

f=P No effect.

SPECIFY COMMAND AND UNIT

DOA<f> ac, CAS

0	1	1	AC		0	1	0	F		0	1	1	1	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 10-15 of the specified AC are loaded into the combined Command/Transport Select Register. Bits 0-9 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

										COMMAND			UNIT		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-9	----	Reserved for future use.
10-12	Command	Select the command for the selected transport as follows: 000 Read 001 Rewind 010 Reserved for future use 011 Space Forward 100 Space Reverse 101 Write 110 Write End Of File 111 Erase
13-15	Unit	Select transport 0-7

READ STATUS

DIA<f> ac, CAS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	0	1	F		0	1	1	1	0	0

The contents of the Status Register are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ERROR	DATA LATE	RE-WIND-ING	ILLE-GAL	1	PAR-ITY ERROR	END OF TAPE	END OF FILE	BEGIN-NING OF TAPE	1	WRITE FAIL	0	0	WRITE LOCK	0	CAS-SETTE UNIT READY

Bits	Name	Meaning When 1
0	Error	One or more of the bits 1, 3, 5, 6, 7, 8 or 10 are 1.
1	Data Late	The data channel failed to respond in time to a data channel request.
2	Rewinding	The selected transport is currently rewinding.
3	Illegal	A Start command was issued to the selected transport when one of the following conditions existed: • The transport was not ready. • The command was Space Reverse and the tape was at the beginning of tape leader. • The command was Write, Write End Of File, or Erase when the tape was write-protected.
4	----	Reserved for future use.
5	Parity Error	The cyclic checkword read from the tape after the record did not match the checkword calculated by the controller.
6	End Of Tape	The transport has reached the end of tape mark. Executing either a Space Reverse or a Rewind operation sets this bit to 0.
7	End Of File	The transport has encountered an End Of File mark in reading or spacing or has just written an EOF mark.
8	Begin Of Tape	The tape is at the beginning of tape mark.
9	----	Reserved for future use.
10	Write Fail	The last record written has caused a failure in the transports write electronics.
11	----	Reserved for future use.
12	----	Reserved for future use.
13	Write Lock	The DGC cassette on the selected transport is write protected.
14	----	Reserved for future use.
15	Cassette Unit Ready	The selected transport and cartridge is ready for use.

LOAD MEMORY ADDRESS COUNTER

DOB<f> ac, MTA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	0	0	F		0	1	1	1	0	0

Bits 1-15 of the specified AC are loaded into the Memory Address Counter. Bit 0 is ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	MEMORY ADDRESS														

Bits	Name	Function
0	----	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

LOAD WORD COUNTER

DOC<f> ac, MTA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	1	0	F		0	1	1	1	0	0

Bits 4-15 of the specified AC are loaded into the controller's Word Counter. Bits 0-3 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The specified AC must contain the two's complement of the number of words to be transferred with the stipulation that the minimum number of words to be transferred is 2 and the maximum is 4096.

During a spacing operation, the Word Counter acts as a record counter. The specified AC must contain the two's complement of the number of records to be skipped with the stipulation that the minimum number of records to be skipped is 1 and the maximum is 4096. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	– WORD COUNT											

Bits	Name	Function
0-3	----	Reserved for future use.
4-15	Word Count	Two's complement of number of words to be transferred or records to be skipped.

READ MEMORY ADDRESS COUNTER

DIB<f> ac, MTA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	1	F		0	1	1	1	0	0

The contents of the Memory Address Counter are placed in bits 1-15 of the specified AC. AC bit 0 is set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. When the Memory Address Counter is read after a Read or a Write operation, the contents point to a memory location one greater than the location of the last word transferred. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	MEMORY ADDRESS														

Bits	Name	Function
0	----	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

PROGRAMMING

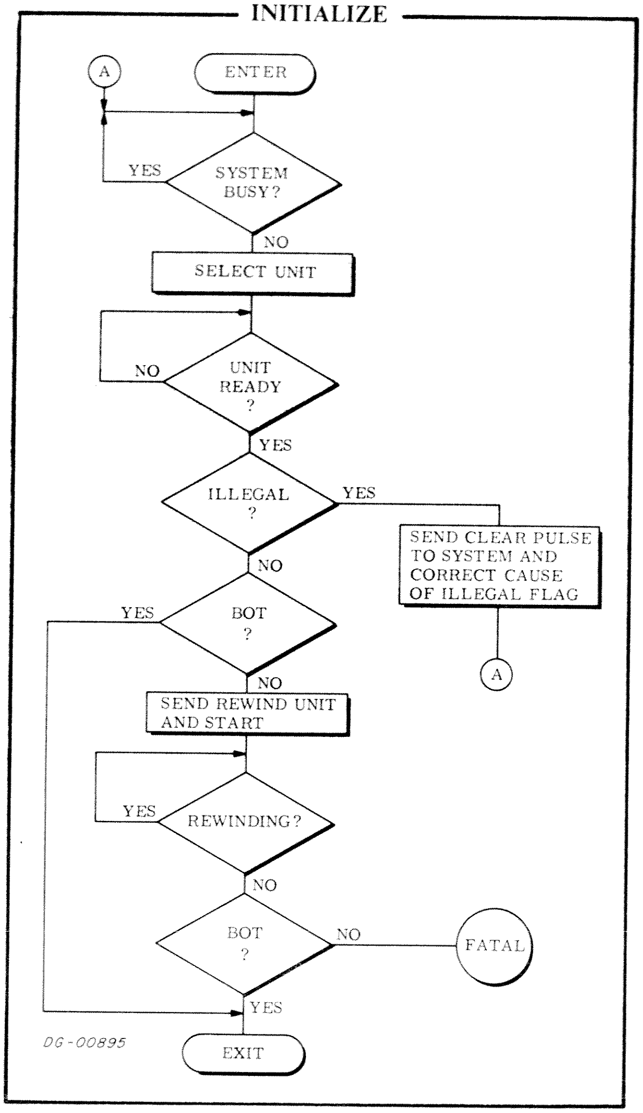
The preparation of a DGC cassette subsystem for a data channel transfer can be divided into three distinct phases: I, initializing the transport; II, positioning the tape; and III, specifying the transfer parameters. Once the first two phases are completed, the program can repeatedly execute the third phase. The results of issuing commands in each phase should be checked for errors before proceeding to the next phase.

Phase I: Initializing

Initialization of the transport is performed as follows: the desired transport is selected by issuing a SPECIFY COMMAND AND UNIT instruction (DOA) to the controller. The Status Register is examined by a READ STATUS instruction (DIA) to determine if the tape is positioned at the Beginning Of Tape mark, if the Illegal flag is set to 1, and if the transport is in the ready state. The transport should be initialized by issuing a Rewind command if the tape is not located at the Beginning Of Tape mark. The Rewind command can be executed providing the transport is ready and the Illegal flag is set to 0.

If the Illegal flag is set to 1 in the Status Register, a Clear command can be issued. The transport is then selected and a Rewind command specified by issuing a SPECIFY COMMAND AND UNIT instruction. The operation is initiated by issuing a Start command. The Start command sets the Rewinding flag to 1 and the Unit Ready flag to 0, but does not set either the Busy flag or the Done flag to 1. During this time, the program can issue instructions to any other transport which is in the ready state. The controller does not initiate a program interrupt request when the rewinding operation is completed. However, the Rewinding flag will be set to 0 and the Unit Ready flag to 1 upon completion of the operation.

Once the tape has been rewound and no error conditions are detected in the Status Register, the program may proceed to Phase II.



Phase II: Position the Tape

In order to access the desired record on the DGC cassette for a Read or a Write operation, the tape should be spaced to the position immediately preceding that record. During the spacing operation, the Word Counter is used as a record counter, incrementing once for each record skipped. The spacing operations terminates when the Word Counter overflows or when an End Of File mark is encountered. The minimum number of records which can be skipped in one operation is one and the maximum is 4096.

The spacing operation is performed as follows: the command for the spacing operation desired is loaded into the Command Register with a SPECIFY COMMAND AND UNIT instruction (DOA), and the two's complement of the number of records to be skipped is specified with a LOAD WORD COUNTER instruction (DOC). The spacing operation is initiated with a Start command appended to the last of these instructions issued. The Busy flag is set to 1 and the Done flag is set to 0 during spacing. Upon completion of the operation, the Busy flag is set to 0 and the Done flag is set to 1, thus initiating a program interrupt request.

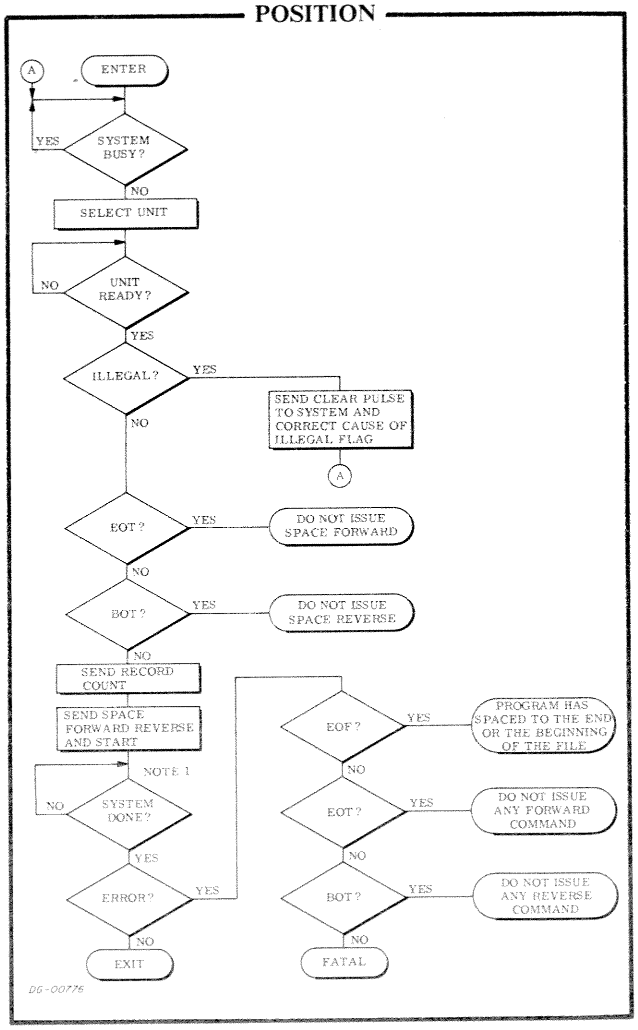
If the desired record is contained in another file, the Space Forward command must be issued at least once for each file to be skipped. When a file is to be skipped, the Word Counter must be loaded with the two's complement of a count greater than or equal to the number of records contained in that file. Since a file may contain more than 4096 records, the Space Forward command may have to be executed several times (reloading the Word Counter each time) before the file is skipped. Every time an End Of File mark is encountered in a spacing operation, the operation stops, both the End Of File flag and the Error flag are set to 1 in the Status Register, the Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

Once the transport has reached the correct file, the tape should be positioned immediately before the desired record in that file.

The Space Reverse command operates in a manner entirely analogous to the Space Forward command. After a Space Reverse command is executed as many times as is necessary to skip a file, the tape is positioned immediately after the last record in the preceding file.

Care should be taken so that a Space Forward command is not executed when the End Of Tape flag is set to 1 in the Status Register.

Once the tape is in position for a Read or a Write operation and no errors (other than the End Of File, Error combination) are detected in the Status Register, the program may proceed to Phase III.


POSITION
A
ENTER
SYSTEM BUSY?
YES
NO
SELECT UNIT
UNIT READY?
NO
YES
ILLEGAL?
YES
SEND CLEAR PULSE TO SYSTEM AND CORRECT CAUSE OF ILLEGAL FLAG
A
NO
EOT?
YES
DO NOT ISSUE SPACE FORWARD
NO
BOT?
YES
DO NOT ISSUE SPACE REVERSE
NO
SEND RECORD COUNT
SEND SPACE FORWARD REVERSE AND START
NOTE 1
SYSTEM DONE?
NO
YES
ERROR?
YES
NO
EXIT
EOF?
YES
PROGRAM HAS SPACED TO THE END OR THE BEGINNING OF THE FILE
NO
EOT?
YES
DO NOT ISSUE ANY FORWARD COMMAND
NO
BOT?
YES
DO NOT ISSUE ANY REVERSE COMMAND
NO
FATAL
DG-00776

Phase III: Specify Transfer Parameters

Phase III consists of specifying the starting location in memory for the first word to be transferred, loading the Word Counter with the two's complement of the number of words to be transferred and issuing either a Read or a Write command together with a Start command to the desired transport.

Read

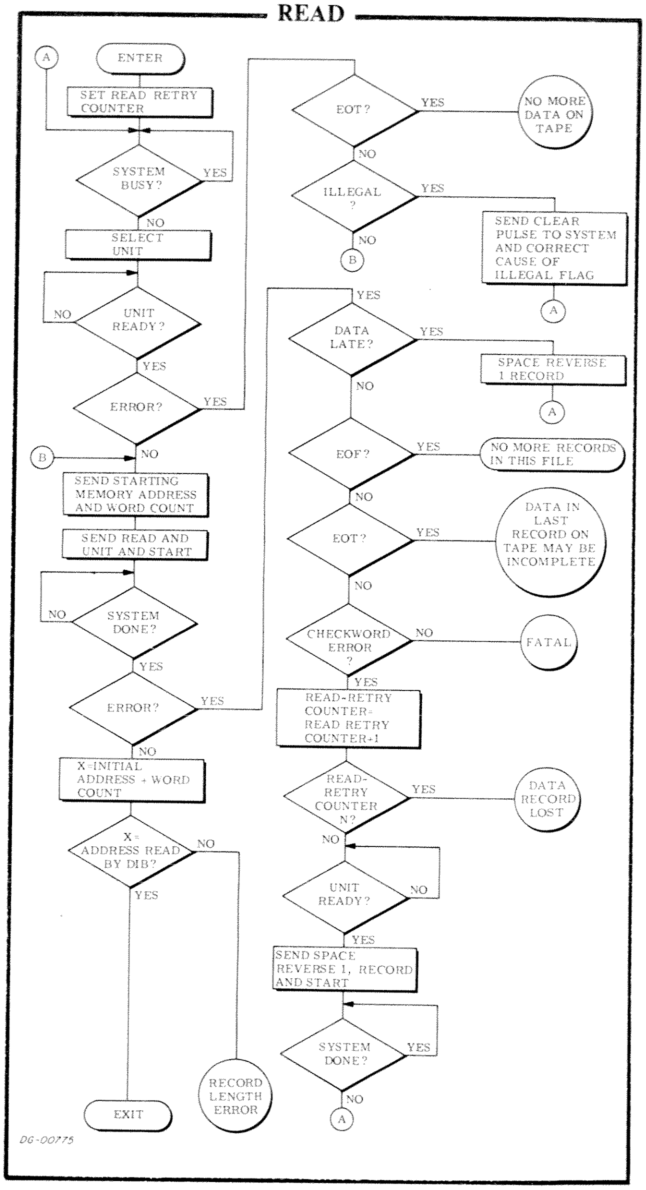
Read operations are performed as follows: the storage location in memory for the first word to be read from the tape is specified by a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The two's complement of the number of words to be read is then specified by a LOAD WORD COUNTER instruction (DOC). If the record to be read is of unknown length, giving the two's complement of the longest possible record, assures that the entire record will be read. The desired transport is selected and the Read command is loaded into the Command Register with a SPECIFY COMMAND AND UNIT instruction (DOA). A Start command in the last of these instructions initiates the Read operation.

Once the Read command is initiated, the Busy flag is set to 1, the Done flag is set to 0, the tape in the DGC cassette is moved past the heads, and the first 16 bits in the record are assembled into one word. The word is then written into memory via the data channel. Each time a word is transferred to memory, the Word Counter is incremented by one.

The Read operation will continue until either the Word Counter overflows or the transport encounters an End Of Record gap. Upon completion, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. If the record read was of unknown length and was read by loading a zero (the two's complement of 4096) in the Word Counter, the length can be determined by first obtaining the address of the last word read into memory. This address is obtained by issuing a READ MEMORY ADDRESS COUNTER instruction (DIB). The address returned is one greater than the address of the last word written in memory by the Read operation. Therefore, subtracting the starting memory location from the address returned by a READ MEMORY ADDRESS COUNTER instruction will yield the length of the record read.

Whenever the End Of Tape flag is set to 1 in the Status Register, care should be taken to ensure that any command which moves the tape forward is not issued.

READ



Write

Write operations (Write, Write End Of File, and Erase) are performed as follows: for a Write command, the storage location in memory for the first word to be written on the tape is specified by a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The two's complement of the number of words to be written is then specified by a LOAD WORD COUNTER instruction (DOC). The desired transport is selected and the Write command is loaded into the command register with a SPECIFY COMMAND AND UNIT instruction (DOA). A Start command in the last of these instructions issued initiates the Write operation.

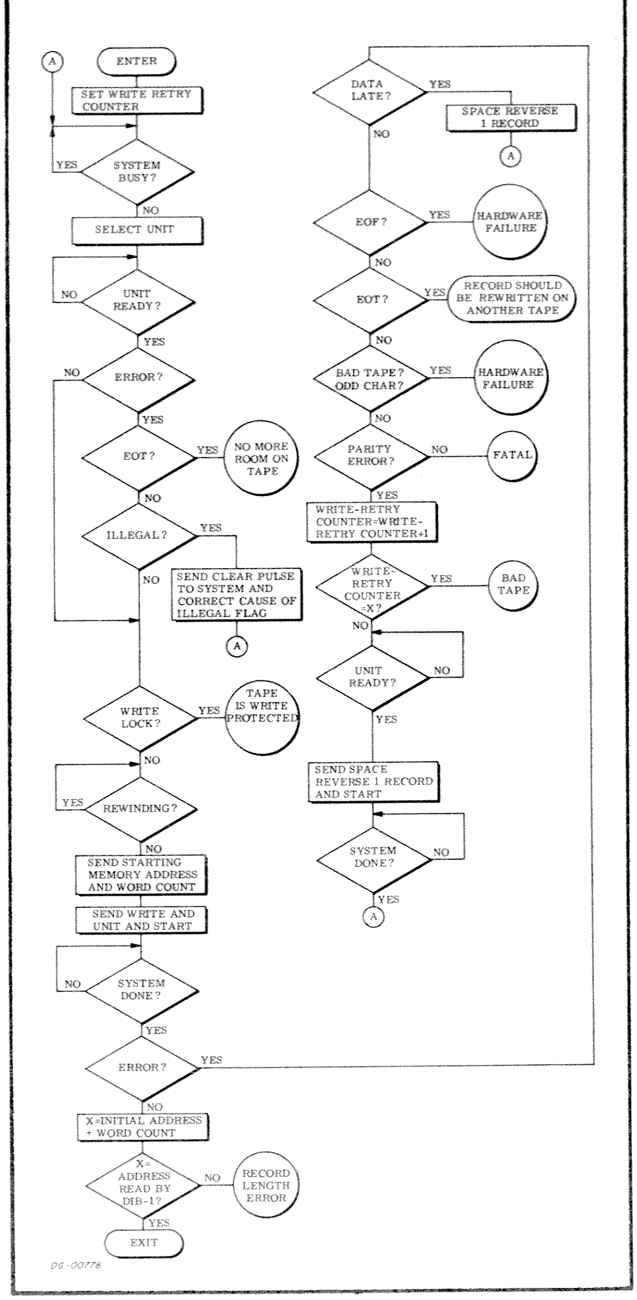
Once the Write command is initiated, the Busy flag is set to 1, the Done flag is set to 0, a word is read from memory, the tape in the DGC cassette is moved past the heads, and the 16 bits of the word are written on the tape. Each time the controller receives a word from memory, the Word Counter is incremented. Once the Word Counter overflows, the controller writes the cyclic redundancy checkword. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Write End Of File and Erase commands are accomplished by selecting the desired transport and loading the desired command into the Command Register with a SPECIFY COMMAND AND UNIT instruction. A Start command sets the Busy flag to 1, the Done flag to 0, and initiates the operation.

The Write End Of File command erases 2 1/2 inches of tape and writes an End Of File mark on the tape after the inter-record gap. Upon completion of this operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. Since the transports perform a read-after-write error check, both the End Of File flag and the Error flag are set to 1 in the Status Register each time a correct End Of File mark is written.

The Erase command erases approximately 2 1/2 inches of tape each time it is issued. Erase can be used to skip bad sections of tape. Once the Erase operation has been completed, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

WRITE



TIMING

The DGC cassette transfers one 16-bit word via the data channel every 1.34 milliseconds when performing a read or a write operation. Since the DGC cassette is a single buffered device, the data channel has only 28 microseconds to respond to a data channel request. If the data channel does not respond within this time, the Data Late and the Error flags are set to 1. When the Data Late flag is set to 1, the processing of the record continues until its normal completion. However, one or more of the words transferred contains an error.

Since the tape does not move at a constant speed because of the recording technique used for the DGC cassette transport, all references to tape speed, inter-record gaps, erased areas of tape, and information density, pertain to the average values along the length of the DGC cassette's tape. On the average, the tape moves past the heads at 30 inches per second when reading, writing, rewinding and spacing. The average bit density on the tape is 430 bits per inch.

The start and stop times for the basic DGC cassette operations are given in the table below.

	Read	Write	Space	Rewind
Start Delay	.22ms	35ms	1.3ms	2sec
Last Word To Stop	136ms	136ms	120ms	2sec

ERROR CONDITIONS

During Initializing

If a Start command is given when the selected unit is not ready to carry out a command, both the Illegal flag and the Done flag are set to 1 and a program interrupt request is initiated. The Illegal flag can be set to 0 only with a Clear command or an I/O RESET instruction. The Clear command sets all the error indicating flags in the Status Register to 0 and sets the Command/Transport Select Register to 0. Therefore, unit 0 is the selected transport and Read is the specified command after a Clear command or an I/O RESET instruction is issued. The Status flags read by a READ STATUS instruction pertain to unit 0 until another transport is selected.

During Positioning

Both the Illegal flag and the Error flag are set to 1 in the Status Register when a Spacing command is issued to a transport which is not ready or if a Space Reverse command is issued when the tape is at the BOT mark. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

If a Space Forward command is issued and the tape is at the End Of Tape mark, the spacing operation continues until a Clear command is issued.

During Reading

If the data channel does not respond in time for a data channel request, both the Data Late flag and the Error flag will be set to 1 in the controller's Status Register, but the Read operation will continue until either the Word Counter overflows or the transport encounters an End Of Record gap on the tape. The Data Late flag indicates that at least one word on the tape was not correctly transferred to memory.

If the transport encounters an End Of File mark, the Error flag, together with the End Of File flag, is set to 1 in the controller's status register. Busy is set to 0, Done is set to 1 and a program interrupt request is initiated.

Once the transport encounters the end of the data in a record or the word counter overflows, the controller reads the checkword and if the checkword does not match that calculated by the controller, the Checkword Error flag is set to 1 in the Status Register. The correct checkword for the entire record will be read even if the program reads only a portion of the record.

During Writing

If the selected transport is not ready to receive a command or if the DGC cassette mounted on the selected transport is write-protected, the Start command will set the Error flag, the Illegal flag and the appropriate error condition flag in the Status Register to 1. Busy will be set to 0, Done will be set to 1 and a program interrupt request is initiated.

If the data channel does not respond in time for a data channel request, both the Data Late flag and the Error flag will be set to 1 in the Status Register but the Write operation will continue until the Word Counter overflows. The Data Late flag indicates that at least one word was not properly written.

If the End Of File mark contains an error, the controller will interpret the mark as a short data record and a Checkword Error will occur on every subsequent Read operation of that section of tape. A defective End Of File mark will also be interpreted as a data record during spacing operation.

Since the DGC cassette performs a read-after-write error check, the Checkword Error flag is set to 1 if the record just written contains an error.

This page intentionally left blank

SECTION V

DISCS

- FIXED HEAD DISC SUBSYSTEM
- 4047A & 4047B DISC CARTRIDGE SUBSYSTEMS
- 4048A DISC PACK SUBSYSTEM
- 4057A DISC PACK SUBSYSTEM
- 4231A DISC PACK SUBSYSTEM

This page intentionally left blank

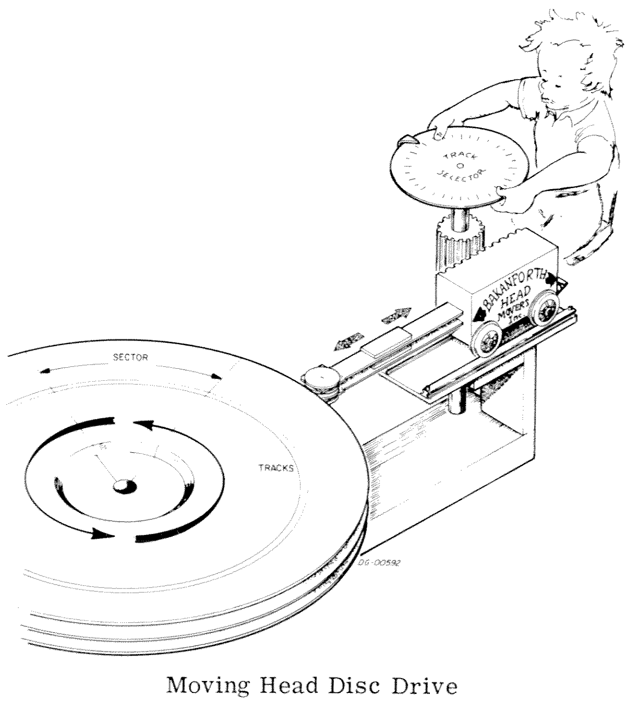
INTRODUCTION TO DGC DISCS

Disc drives are popular for storing large quantities of information which must be directly accessed. The basic recording medium is a magnetic material coated on a platter. Platters come either singly or in a stack. The information on the platters is recorded or read by heads suspended near their surfaces. As the platters rotate, the heads define concentric circles of data called tracks around the surfaces of the platters. Heads can be either fixed or moving. Fixed-head discs assign one head to each and every operating track, while moving-head discs use one head for each surface so each head moves back and forth to cover all tracks.

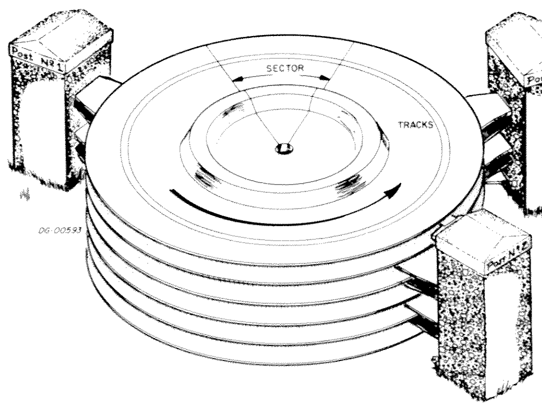
Heads are mounted on arms to extend them out over the recording surfaces. The read/write heads together with their arms form an assembly called the access mechanism. The access mechanism on a multi-surface moving-arm disc consists of a number of arms mounted on a single post and extended like a comb between the platters. This entire mechanism moves in and out as single unit when the heads are being positioned.

The access mechanism on a multi-surface fixed-head disc also consists of a number of arms mounted on a post but the assembly does not move during routine operation of the disc. Each arm usually has many heads attached to it; each head is located over its respective track. Several posts can be located around the peripheral of the stack of platters; each post then carries its own complement of arms and heads.

The platters on some types of disc drives can be removed and exchanged. Removable single platters are called disc cartridges, removable stacks are called disc packs.



Moving Head Disc Drive



Fixed Head Disc Drive

DATA FORMATS ON DGC DISCS

Data General stores data on all its disc drives in a standard format. Data is recorded serially, bit by bit, on one track at a time. Sixteen bits form a word, and 256 contiguous words form a data storage field called a sector. A sector is the smallest addressable unit of information. The number of sectors which are recorded on each track is dependent on the particular disc drive used.

In addition to the data contained in each sector, a 16-bit cyclical checkword is recorded at the end of the data field. This checkword is calculated as the individual bits of the sector are passed from the controller to the drive, to be written on the disc. Once the 256 words (4096 bits) in the record have been written, this checkword is written. Subsequently, when that sector is read, the controller again calculates a checkword from the individual bits read from the disc. This new checkword is then compared to the original checkword as it is read from the disc at the end of the sector. If the two checkwords differ, an error flag is set to 1, indicating that the data in the sector just read may contain errors.

On some discs, each sector also contains an identification field which precedes the data field. This identification field is used to verify the location of the sector before any read or write operation is performed and, in some cases, to indicate that the sector should not be used because the recording surface is defective.

The process of writing these identification fields is called formatting. All discs must be properly formatted with special programs provided by Data General before they can be used. The part numbers for these programs are given in the section dealing with the programming for each disc.

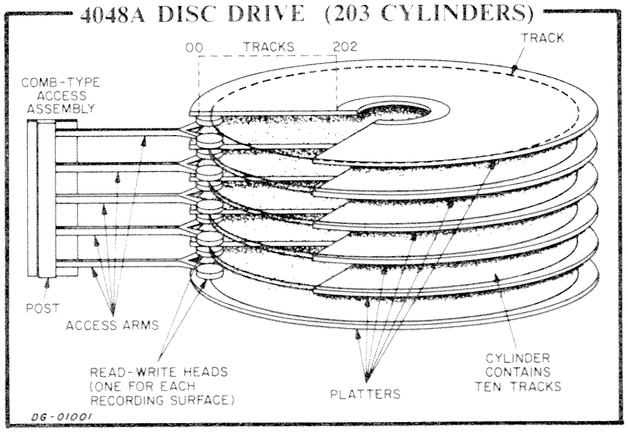
ACCESSING

The data stored in any particular sector of a fixed-head disc is accessed by first selecting the read/write head which is assigned to the track in which the sector is located and then waiting until the desired sector passes under that head. Maximum access time is therefore the sum of the times required for head selection, (on the order of 1 millisecond) and the maximum time the head must wait until the beginning of the desired sector appears, i.e., one complete revolution of the platter (17 milliseconds @ 3540rpm).

The data stored in any particular sector of a moving-head disc is accessed by first moving the access mechanism to the track which contains the sector, selecting the head for the proper track, and then waiting until the desired sector passes under that head. Maximum access time is therefore the sum of the time required to move the access mechanism (this is called a Seek operation and required from 10 to 135 milliseconds) plus the time required to select the head (this time can overlap the time required for head positioning) plus the maximum time the head must wait until the beginning of the desired sector to appear (this time could be up to one or two complete revolutions of the platter, depending on the particular units synchronizing mechanism. The range is from 17 to 48 milliseconds.)

However, the average access time for a set of sectors on a moving-arm disc can be reduced because the access mechanism positions all the heads simultaneously. Once the access mechanism is locked in place, each head can access one complete track without being repositioned. This set of accessible tracks is called a "cylinder". In other words, there are as many cylinders in a disc as there are discrete positions (tracks) of the access mechanism over the surface and as many tracks in each cylinder as there are recording surfaces.

4048A DISC DRIVE (203 CYLINDERS)


00 TRACKS 202
TRACK
COMB-TYPE ACCESS ASSEMBLY
POST
ACCESS ARMS
READ-WRITE HEADS (ONE FOR EACH RECORDING SURFACE)
PLATTERS
CYLINDER CONTAINS TEN TRACKS
DG-01001

DGC DISC SUBSYSTEMS

Data General offers one fixed-head disc subsystem which uses any one of four available drive units, and five distinct moving-head disc subsystems each of which require their own particular drive units. The following table lists the specifications, equipment requirements and maximum sizes of the subsystems.

Specifications for DGC Disc Subsystems

Subsystem	Controller	Adapter	Drive Unit	Type	Number of Tracks or Cylinders	Number of Surfaces	Number of Sectors/Track	Capacity/ Unit (Words)	Transfer Rate (Words/Sec)	Maximum Number of Units/Subsystem
Fixed-head	4019	None Required	6001/6005	1 Post	64 Tracks	8*	8	131,072	122,700	1024 Tracks In Any Combination
			6002/6006	1 Post	128 Tracks		8	262,144		
			6003/6007	2 Post	256 Tracks		8	524,288		
			6004/6008	3 Post	384 Tracks		8	786,432		
4047A and 4047B	4046	4049	4047A	Cartridge	203	2	12	1,247,232	90,000	4
		4047	4047B	Cartridge and Nonremovable	203+203	2+2	12	2,494,464		2
4048A	4046	4048	4048A	Pack	203	10	6	3,118,080	78,000	4
4057A	4046	4057	4057A	Pack	203	20	12	12,472,320	156,000	4
4231A	4231	4231A (comes with first drive)	first drive 4231A	Pack	411	19	23	45,979,392	403,000	4
			second-fourth drive 4231B							

*The number of surfaces in a fixed head disc is transparent to the programmer.

SHARED DISC CONSIDERATIONS

The following sections on programming the various disc subsystems all assume that the disc drive is directed by one CPU. However, all disc subsystems can be shared by two CPU's. In the case of the Fixed Head Disc Subsystems, no change in either the instructions themselves or the order in which they are issued is required. Access to the drive is alternated between the two CPU's if there is a conflict. As a result of sharing the subsystem, the time for accessing consecutively numbered sectors by one CPU is extended to 10.5 milliseconds/sector (average).

In the case of a Moving Head Disc Subsystem, the adapter performs certain functions which are no longer transparent to the programmer. These functions force a different instruction sequence for programming a data channel transfer. Each subsystem adapter alternates access between the two CPU's. When one of the CPU's starts an operation (seeking, reading or writing), the adapter locks out the second CPU until either the Read/Write Done flag is set to 1 for the first CPU or six seconds elapse. When a CPU is locked out by the adapter, that CPU can neither select a drive unit nor read a valid status word for any drive.

The six second lock out procedure insures that one CPU cannot seize the disc subsystem to the exclusion of the second CPU. This prevents a malfunctioning CPU from interferring with the operation of the second CPU for more than six seconds.

In addition, since either CPU can position the heads on any drive, the position of the heads in the selected drive is unknown to a CPU when it gains access to that drive. Therefore, each data transfer operation should be preceded by a Seek operation to the desired cylinder on the selected drive. If the adapter is in use when this Seek operation is initiated, the operation commences as soon as the adapter becomes free, and the other CPU is then locked out. When the Seek Done flag for the selected drive is set to 1 and a program interrupt request is initiated, the program can read a valid status word and perform a Read or a Write operation. As soon as the Read/Write Done flag is set to 1, the adapter is free to service any requests from either CPU.

Although the adapter might transfer control to the other CPU once the Read/Write Done flag is set to 1, the error indicating flags in the status register of the first computer are valid for the operation just completed. These flags are: Error, Data Late, Check Error, Address Error, End Error and the Sector Error, Head Error, and Bad Sector flags, where applicable.

One situation which can be encountered in a shared disc environment is that of losing control of the adapter during an operation due to the 6 second time out provision. The six second time interval is measured from the start of the Seek operation in the drive itself. If the program does not initiate the data transfer operation so that is has sufficient time to be completed before 6 seconds elapse, the operation will not be completed. This situation could exist if a seek error occurs and the drive is recalibrated several times before a Seek operation is successful. Then, a data transfer operation might not have enough time to be completed before the 6 seconds elapses.

A second situation could exist in that a drive might be performing a Seek or Recalibrate operation, after a seek error, when the six seconds elapse. The computer that was locked out could initiate its own Seek operation with the same drive. This condition could cause damage to the access mechanism of the drive unit.

In order to avoid ata loss or damage to a drive, the program should recalibrate the drive unit after a seek error and perform a "dummy" Read operation. This dummy read should read one 256 word sector from cylinder 0 into a scratch buffer. Once the operation is completed, the adapter is free. The program can then attempt a second Seek operation to the desired cylinder and have 6 seconds to complete the data transfer.

If, after several attempts to recalibrate the drive, seek errors still occur, the drive unit should be considered inoperative. No attempts should be made to access that drive until it is repaired.

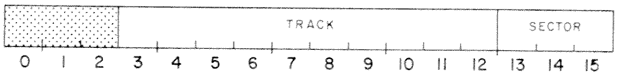
FIXED HEAD DISC SUBSYSTEM

SUMMARY

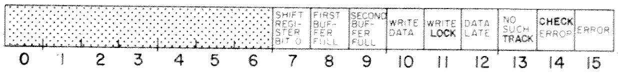
MNEMONIC (FIRST CONTROLLER)	DSK
DEVICE CODE (FIRST CONTROLLER)	20_8
MNEMONIC (SECOND CONTROLLER)	DSK1
DEVICE CODE (SECOND CONTROLLER)	60_8
PRIORITY MASK BIT	9
TRACKS	64-1024
SECTORS/TRACK	8
WORDS/SECTOR	256
TOTAL STORAGE CAPACITY MIN/MAX (WORDS)	131,072/2,097,152
MAXIMUM TRANSFER RATE (WORDS/SEC)	122,700
ALLOWABLE DATA CHANNEL LATENCY (μSEC)	14
SECTOR ACCESS TIME MAX/MIN (MSEC)	18/1

ACCUMULATOR FORMATS

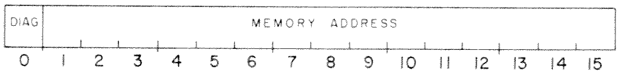
SPECIFY DISC TRACK AND SECTOR... (DOA)



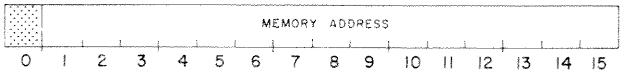
READ STATUS........................ (DIA)



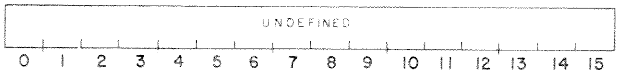
LOAD MEMORY ADDRESS COUNTER ...(DOB)



READ MEMORY ADDRESS COUNTER....(DIB)



DIAGNOSTIC........................ (DIC)



S, C AND P FUNCTIONS

S Set Busy to 1, Done and all status flags to 0 and start a Read operation.

C Set Busy, Done, Data Late and Check Error flags to 0 and stop all operations.

P Set Busy to 1, Done and all status flags to 0 and start a Write operation.

INTRODUCTION

The 6001-6008 drive units are used in the fixed head disc subsystem. These differ only in storage capacity (number of tracks). The minimum number of tracks in a subsystem is 64 ($0-77_8$) while the maximum is 1024 ($0-1777_8$). Each track contains 8 (0-7) sectors; each of which stores 256 (400_8) 16-bit words together with a checkword for the sector. The data storage capacity is 2048 words/track. Therefore, a fixed head disc subsystem can store between 131,072 words (for the smallest configuration) and 2,097,152 words (for the largest configuration).

The average access time for any sector in the subsystem is 8.5 milliseconds. Once the sector is found, all data transfers to and from the subsystem occur at a rate of 122,700 words/second. Each transfer operation moves one 256 word sector.

Data, stored in the subsystem, can be protected from accidental overwrite by means of an operator controlled write-protection feature. This allows the operator to write-protect groups of 16 (20_8) tracks.

INSTRUCTIONS

The disc drive controller contains three program accessible registers: a 15-bit Memory Address Counter, a 9-bit Status Register and a 13-bit combined Track/Sector Select Register. The Memory Address Counter is self-incrementing and contains the memory location of the next 16-bit word to be read from or written on the disc. The Status Register contains all the information flags for the disc drive. The combined Track/Sector Select Register contains the number of the desired track on the disc and the number of the desired sector which is to be read or written. There is no Word Counter available to the programmer since the data is always transferred in 256 word blocks.

Four instructions are used to program data channel transfers to and from the fixed head disc subsystem. Two of these instructions are used to supply all of the necessary data to the controller for any disc operation. The remaining two instructions allow the program to determine, in detail, the current state of the subsystem. A fifth instruction is used for maintenance purposes.

The disc controller's Busy and Done flags are controlled using all three of the device flag commands as follows:

f = S	Set the Busy flag to 1, the Done flag and all the status flags to 0, and initiate a Read operation.
f = C	Set the Busy flag, the Done flag, the Check Error and Data Late flags to 0, and terminate any Read or Write operation in progress.
f = P	Set the Busy flag to 1, the Done flag and all status flags to 0, and initiate a Write operation.

SPECIFY TRACK AND SECTOR

DOA<f> ac, DSK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	0	F		0	1	0	0	0	0

Bits 3-15 of the specified AC are loaded into the disc controller's Track/Sector Select Register. Bits 0-2 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨			TRACK										SECTOR		

Bits	Name	Function
0-2	----	Reserved for future use.
3-12	Track	Specify the track number 0-1777_8.
13-15	Sector	Select the sector number 0-7_8 for a Read or a Write operation.

LOAD MEMORY ADDRESS COUNTER

DOB<f> ac, DSK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	0	0	F		0	1	0	0	0	0

Bits 0-15 of the specified AC are loaded into the disc controller's Memory Address Counter. If AC bit is 1, the controller is put into the diagnostic mode of operation. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DIAG	MEMORY ADDRESS														

Bits	Name	Function
0	Diag	Places the controller in the diagnostic mode of operation.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

READ STATUS

DIA<f> ac, DSK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	0	1	F		0	1	0	0	0	0

The contents of the Status Register are placed in bits 7-15 of the specified AC. Bits 0-6 are set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
							SHIFT REGISTER BIT 0	FIRST BUFFER FULL	SECOND BUFFER FULL	WRITE DATA	WRITE LOCK	DATA LATE	NO SUCH TRACK	CHECK ERROR	ERROR

Bits	Name	Meaning When 1
0-6	----	Reserved for future use.
7	Shift Register Bit 0	Used for maintenance.
8	First Buffer Full	Used for maintenance.
9	Second Buffer Full	Used for maintenance.
10	Write Data	Used for maintenance.
11	Write Lock	The Write operation specified a track which was Write-Protected.
12	Data Late	The data channel failed to respond in time to a data channel request.
13	No Such Track	The track specified by the program is not connected to the controller.
14	Check Error	The checkword read from the disc does not match the checkword calculated by the controller.
15	Error	One or more of the bits 11-14 is set to 1.

READ MEMORY ADDRESS COUNTER

DIB<f> ac, DSK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	1	F		0	1	0	0	0	0

The contents of the Memory Address Counter are placed in bits 1-15 of the specified AC. Bit 0 is set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. When the Memory Address Counter is read after a Write operation, the contents point to a memory location two greater than the location of the last word written on the disc. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	MEMORY ADDRESS														

Bits	Name	Function
0	----	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

DIAGNOSTIC

DIC<f> ac, DSK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	0	1	F		0	1	0	0	0	0

When the controller has been placed in the maintenance mode of operation by a 1 in bit 0 of the specified AC in a LOAD MEMORY ADDRESS COUNTER instruction, a DIAGNOSTIC instruction will supply a single clock pulse to the controller's logic. After the clock pulse is supplied, the controller's Busy and Done flags are set according to the function specified by F. Examples of the use of this instruction can be found in the Fixed Head Disc Diagnostic Program Listing (DGC #097-000012).

PROGRAMMING

The preparation of the fixed head disc subsystem for a data channel transfer is divided into two distinct phases: I, specifying the parameters of the transfer; and II, initiating the data transfer. The results of issuing instructions in Phase I should be checked for errors before proceeding to Phase II.

Phase I: Specify the Parameters of the Transfer

Phase I consists of issuing two instructions. They can be executed in any order. Issuing a SPECIFY TRACK AND SECTOR (DOA) instruction to the controller selects the desired track and sector to be processed. A LOAD MEMORY ADDRESS COUNTER (DOB) is then issued to specify the first location in memory to be used in the data channel transfer. Bit 0 of the specified AC must be 0 in this instruction if the subsystem is to perform a Read or Write operation.

When the program is executing successive read operations into contiguous areas of memory, the Memory Address Counter does not have to be updated after each sector is read. However, if contiguous areas of memory are to be written on the disc, the Memory Address Counter must be updated after each Write operation. This is necessary since the Memory Address Counter points to a location in memory which is two greater than the memory location of the last word written on the disc.

Once the track is specified, the status must be checked to determine if the subsystem is ready to proceed. The status is checked by examining the Busy flag of the subsystem. If the Busy flag is 0, the program can proceed to Phase II.

Phase II: Initiate the Transfer

Phase II consists of issuing either a Read or a Write command to the disc subsystem. The Read command transfers a sector (block) of data, consisting of 256 words, from the disc to the computer's memory via the data channel. A Write command transfers a block of data from the computer's memory via the data channel, and stores the data on the disc. Each of these commands is initiated by issuing one of the device flag commands. A Start command initiates a Read operation while a P command initiates a Write operation. Either of these two commands can be appended to the last of the two instructions issued in Phase I.

Read

When a Read command is issued, the Busy flag is set to 1 and the Done flag is set to 0. The controller selects the specified track and then waits until the desired sector is encountered. As the sector passes under the head, the sequential bits are read. When a word is fully assembled, the controller transfers the word to the computer's memory via the data channel. Each time a word is transferred to memory, the Memory Address Counter is automatically incremented.

Once the 256 words have been read, the controller reads the checkword at the end of the sector and compares it to the checkword it had calculated during the read operation. If the two checkwords differ, both the Error flag and the Checkword Error flag are set to 1. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

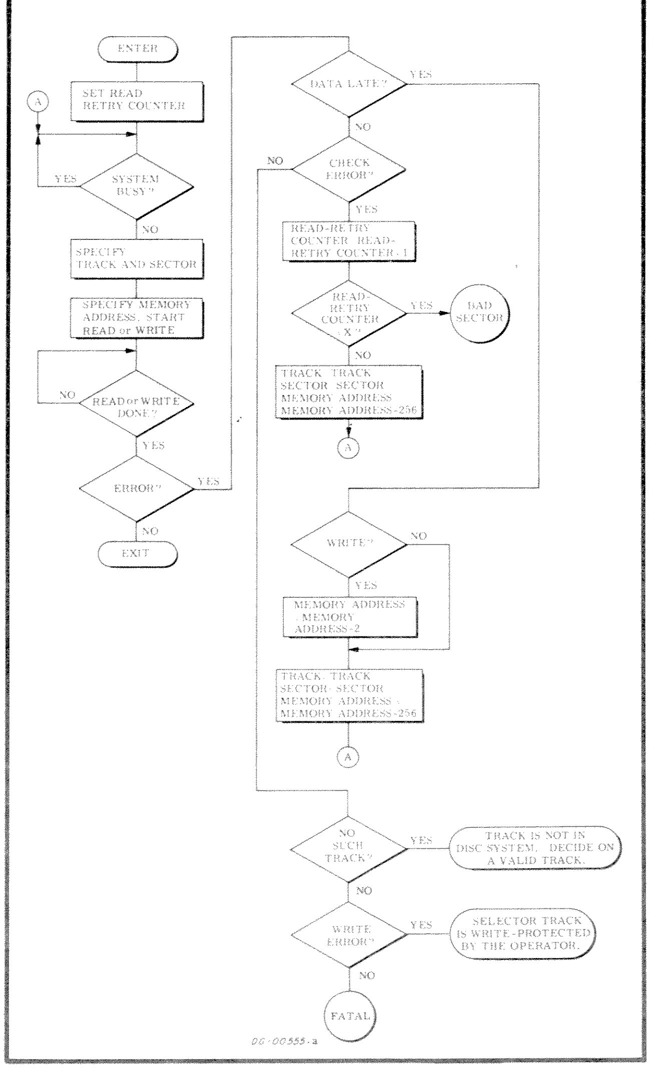
Write

When a Write command is issued, the Busy flag is set to 1 and the Done flag is set to 0. The controller reads three words from the computer's memory, via the data channel, and then waits for the desired sector to pass under the head. Each time the controller reads a word from the computer's memory, the Memory Address Counter is automatically incremented.

Once the desired sector is encountered, the bits of each word are sequentially written. When the 256 words of the sector have been written, the controller writes the checkword it calculated from the data during the Write operation. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

The last two words requested from the computer's memory are not written. This means that the Memory Address Counter points to a memory location which is two greater than the address of the last word written on the disc when the Write operation is concluded.

TRANSFER ONE SECTOR


ENTER
SET READ RETRY COUNTER
A
SYSTEM BUSY?
YES
NO
SPECIFY TRACK AND SECTOR
SPECIFY MEMORY ADDRESS, START READ or WRITE
READ or WRITE DONE?
NO
YES
ERROR?
YES
NO
EXIT
DATA LATE?
YES
NO
CHECK ERROR?
NO
YES
READ-RETRY COUNTER READ-RETRY COUNTER-1
READ-RETRY COUNTER X?
YES
BAD SECTOR
NO
TRACK TRACK SECTOR SECTOR MEMORY ADDRESS MEMORY ADDRESS-256
A
WRITE?
NO
YES
MEMORY ADDRESS MEMORY ADDRESS-2
TRACK TRACK SECTOR SECTOR MEMORY ADDRESS MEMORY ADDRESS-256
A
NO SUCH TRACK?
YES
TRACK IS NOT IN DISC SYSTEM. DECIDE ON A VALID TRACK.
NO
WRITE ERROR?
YES
SELECTOR TRACK IS WRITE-PROTECTED BY THE OPERATOR.
NO
FATAL
DG-00555-a

TIMING

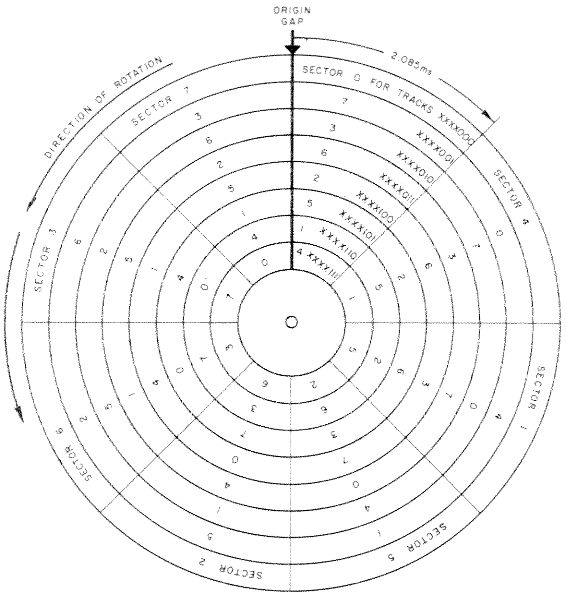
The disc rotates at a speed of 3540rpm. Therefore, the time for a complete revolution is 16. 95 milliseconds. Since there are eight sectors on a track, the time for one sector to pass under the Read/Write head is 2. 12 milliseconds. If the Read or the Write command is given at the same time as the SPECIFY TRACK AND SECTOR instruction, a minimum of 1 milliseconds elapses before the data transfer can begin. This allows the head selection logic to settle down. The maximum time for a sector search is 17. 95 milliseconds so the average sector search latency time is 9. 5 milliseconds.

The data segment of a sector passes under the head in 2. 08 milliseconds which means the data channel requests occur every 8. 15 microseconds. Since the fixed head disc system is double buffered, the maximum allowable data channel latency is 14 microseconds. If the data channel does not respond within this time, the Data Late flag will be set to 1. When the Data Late flag is set to 1, one or more words are lost but the processing of the sector continues and when the Done flag is set to 1, a program interrupt request is initiated.

When the program is processing consecutively numbered sectors, the program has 1. 12 milliseconds after the Done flag is set to 1 to issue a SPECIFY TRACK AND SECTOR instruction to select the next sector, except when the sector just processed is sector 3. The program then has a 3. 24 milliseconds to select sector 4 because both sectors 7 and 0 lie between sectors 3 and 4 on the disc surface. (See DISC FORMAT section.)

DISC FORMAT

The controller cannot process physically adjacent sectors on the disc surface consecutively. In order to minimize waiting time, the sectors on a track are not numbered consecutively but are interleaved in the manner shown below. For example, when the drive finishes processing sector 0 on a track, sector 4 starts passing under the Read/Write head for that track. During the interval that sector 4 is under the head, the program has sufficient time to issue the instructions necessary to process sector 1 which will pass under the head immediately after sector 4. The interleaving of sectors is continued from one track to the next so that having completed processing sector 7 on one track, sector 0 on the next track can be processed with a minimum of time required for sector search. This numbering system is continued over all the tracks in the disc system. Eight possible configurations are under the heads at any time. These configurations of sector numbering repeat for every eight tracks. The particular arrangement is determined by the least significant octal digit in the track address.



A. SECTOR LOCATIONS ON THE DISK

TRACK	ORIGIN GAP	SECTORS							
XX0		0	4	1	5	2	6	3	7
XX1		7	0	4	1	5	2	6	3
XX2		3	7	0	4	1	5	2	6
XX3		6	3	7	0	4	1	5	2
XX4		2	6	3	7	0	4	1	5
XX5		5	2	6	3	7	0	4	1
XX6		1	5	2	6	3	7	0	4
XX7		4	1	5	2	6	3	7	0
SEC CTR		0	1	2	3	4	5	6	7

B. TRACK - SECTOR LOCATIONS RELATIVE TO THE SECTOR COUNTER

ERROR CONDITIONS

During Specification of the Parameters of the Transfer

If the track selected in the SPECIFY TRACK AND SECTOR instruction does not exist in the subsystem, no indication is given until the program attempts to perform a Read or a Write operation. When the program attempts to read or write a non-existent track, the controller terminates the operation. The Error and the No Such Track flags are set to 1; the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The No Such Track flag is set to 0 when the program completes a Read or a Write operation on a valid track.

During a Read Operation

After the 256 words of the sector have been read, the checkword is read and compared with the checkword calculated by the controller from the data during the Read operation. If the checkwords differ, both the Error and the Check Error flags are set to 1. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Check Error flag indicates that at least one of the words read contains an error.

If the data channel fails to respond to a data channel request in the time allowed, both the Error and the Data Late flags are set to 1. The Read operation continues until the end of the sector; the Busy flag is then set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word in the sector was not properly transferred to the computer.

During a Write Operation

If the track specified in Phase I is write-protected, the sector selected to receive the data is not altered. However, the controller performs all the tasks necessary for a Write operation. Words are read from memory and the Memory Address Counter is incremented as in a valid Write operation. Once the attempt to write the sector has concluded, the Busy flag is set to 0, the Error and the Write Lock flags are set to 1, the Done flag is set to 1 and a program interrupt request is initiated. The Write Lock flag is set to 0 when the program completes a Read or a Write operation on a valid track.

If the data channel fails to respond to a data channel request in the time allowed, both the Error and the Data Late flags are set to 1. The Write operation continues until the end of the sector; the Busy flag is then set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word in the sector was not properly written.

THE 4047A AND 4047B DISC CARTRIDGE SUBSYSTEMS

SUMMARY

MNEMONIC (FIRST CONTROLLER)	DKP
DEVICE CODE (FIRST CONTROLLER)	33_8
MNEMONIC (SECOND CONTROLLER)	DKP1
DEVICE CODE (SECOND CONTROLLER)	73_8
PRIORITY MASK BIT	7
SURFACE/UNIT	2
TRACKS/SURFACE (CYLINDERS)	203
SECTORS/TRACK	12
WORDS/SECTOR	256
TOTAL STORAGE CAPACITY/UNIT (WORDS)	1,247,232
MAXIMUM TRANSFER RATE (WORDS/SEC)	90,000
ALLOWABLE DATA CHANNEL LATENCY (μSEC)	22.2
SEEK TIME MAX/MIN (mSEC)	135/15
SECTOR ACCESS TIME MAX/MIN (mSEC)	40.5/.5

ACCUMULATOR FORMATS

SPECIFY DISC ADDRESS AND SECTOR COUNT .... (DOC)

DRIVE		▨					SUR-FACE	SECTOR				-SECTOR COUNT			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

READ DISC ADDRESS .... (DIC)

DRIVE		▨					SUR-FACE	SECTOR				-SECTOR COUNT			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

SPECIFY COMMAND AND CYLINDER... (DOA)

CLEAR DONE FLAG					▨	COMMAND		CYLINDER							
READ/WRITE	SEEK 0	SEEK 1	SEEK 2	SEEK 3											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

COMMANDS

00 Read
01 Write
10 Seek
11 Recalibrate

READ STATUS .... (DIA)

COMMAND FINISHED					SEEKING ON DRIVE				DRIVE READY	SEEK ERROR	END ERROR	UNSAFE	CHECK ERROR	DATA LATE	ERROR
READ/WRITE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	0	1	2	3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

LOAD MEMORY ADDRESS COUNTER ... (DOB)

▨	MEMORY ADDRESS														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

READ MEMORY ADDRESS COUNTER.... (DIB)

▨	MEMORY ADDRESS														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

S, C AND P FUNCTIONS

S Set Busy to one, Done to zero and start a Read or a Write operation.

C Set Busy to zero, Done to zero and stop all operations.

P Set Done to zero and start a Seek or a Recalibrate operation.

INTRODUCTION

The 4047 disc cartridge subsystem drives are available in two configurations: the 4047A consists of a single drive unit with a removable cartridge while the 4047B has two drives in the same package. One of the drives in the package has a removable cartridge; the other drive contains a permanently mounted disc.

Each drive unit contains a single disc with two surfaces, 0 and 1. Thus a cylinder contains two tracks. There are 203 (312_8) cylinders on a disc. Each of the two tracks in a cylinder contains 12 ($0\text{-}13_8$) sectors each of which stores 256 (400_8)

16-bit words and contains a checkword. The data storage capacity is 3072 words/track, 6144 words/cylinder or 1,247,232 words/cartridge. Words are transferred to and from the subsystem via the data channel at a rate of 90,000 words per second. Up to 16 sectors containing 4096 (10000_8) words can be transferred in one operation.

The controller for a disc cartridge subsystem, when coupled to the adapter, can direct the activities of up to four drive units. Any number of these units can be performing Seek operations simultaneously, but only one drive unit can be reading or writing at any one time.

INSTRUCTIONS

The disc drive controller contains four program accessible registers: a 15-bit Memory Address Counter, a 16-bit Status Register, a 16-bit combined Command/Cylinder Select Register and a combined disc Address/Sector Counter Register. The Memory Address Counter is self-incrementing and contains the memory location of the next 16-bit word to be either read from or written on the disc. The Status Register contains all the information flags for the controller and the selected drive and the seek status of the remaining three drives. Five of the flags in the Status Register are able to initiate a program interrupt request when they are set to 1. These are the Read/Write Done flag, and the 4 Seek Done flags for the drive units 0-3, respectively. The combined Command/Cylinder Select Register contains the command last issued to the subsystem and the number of the desired cylinder on the disc surface. The combined Disc Address/Sector Counter contains the surface and sector location of the active head and the two's complement of the number of sectors to be either read from or written on the disc. The Sector Counter is self-incrementing after each sector is read or written.

Six instructions are used to program data channel transfers to and from the disc pack. Three of these instructions are used to supply all of the necessary data to the controller for any disc operation. The remaining three instructions allow the program to determine, in detail, the current state of the disc pack subsystem.

The disc controller's Busy and Done flags are controlled using all three of the device flag commands as follows:

f=S Set the Busy flag to 1, the Done flag and all the error indicating flags to 0, and initiate a Read or a Write operation, depending on the contents of the Command Register. The error indicating flags are the Seek Error, the End Error, the Unsafe, the Check Error, the Data Late, and the Error flags.

f=C Set the Busy, the Done and all error indicating flags to 0 and stop all positioning and data transferring operations.

f=P Initiate either a Seek or a Recalibrate operation, depending on the contents of the Command Register.

SPECIFY DISC ADDRESS AND SECTOR COUNT

DOC <u><f></u> <u>ac</u>, DKP

0	1	1	AC		1	1	0	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-15 of the specified AC are loaded into the Disc Address Register. Bits 3-6 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

DRIVE		FOR	▨				SURFACE	SECTOR				- SECTOR COUNT			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-1	Drive	Select the drive number, 0, 1, 2 or 3.
2	Format	If 1, place the drive in format mode. This mode should not be used for the 4047A and 4047B subsystem.
3-6	---	Reserved for future use.
7	Surface	Select the surface number, 0 or 1, for the start of a Read/Write operation.
8-11	Sector	Select the starting sector number, 0-13_8, for a Read or Write operation.
12-15	-Sector Count	Specify the two's complement of the number of sectors to be processed (maximum of 16).

SPECIFY COMMAND AND CYLINDER

DOA <f> ac, DKP

0	1	1	AC		0	1	0	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-15 of the specified AC are loaded into the Command/Cylinder Select Register. Bit 5 is ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

CLEAR DONE FLAGS															
READ/ WRITE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	▨	COMMAND		CYLINDER							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function if Set to One
0	Clear Read/ Write Done	Set the Read/Write Done flag, the Check Error, the EOC, the Unsafe, and the Seek Error flags to 0.
1-4	Clear Seek Done	Set the Seek Done flags to 0 for the drives 0-3, respectively.
5	---	Reserved for future use.
6-7	Command	Select the command for the selected drive as follows: 00 Read with Start command. 01 Write with Start command. 10 Seek for the cylinder specified in bits 8-15 of this instruction with a Pulse command. 11 Recalibrate by forcing the heads to cylinder 0 on the selected drive with a Pulse command.
8-15	Cylinder	Specify cylinder number 0-312_8 for the selected drive.

LOAD MEMORY ADDRESS COUNTER

DOB <f> ac, DKP

0	1	1	AC		1	0	0	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

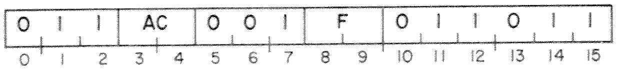
Bits 0-15 of the specified AC are loaded into the controller's Memory Address Counter. Bit 0 must be 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

▨	MEMORY ADDRESS														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

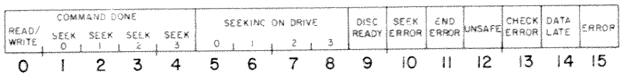
Bits	Name	Contents
0	---	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used in a data channel transfer.

READ STATUS

DIA <f> ac, DKP



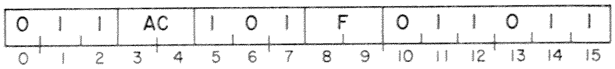
The contents of the Status Register are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:



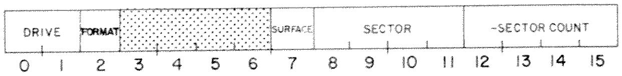
Bits	Name	Meaning When Set to 1
0	Read/Write Done	The selected drive has completed a Read or a Write operation.
1-4	Seek Done	Drive unit 0-3, respectively, has completed a Seek or Recalibrate operation.
5-8	Seeking on Drive	Drive unit 0-3, respectively, is currently performing a Seek or Recalibrate operation.
9	Drive Ready	The selected drive is ready to carry out a command.
10	Seek Error	The selected drive did not successfully carry out the Seek or Recalibrate ordered.
11	End Error	The selected drive attempted to continue a Read or a Write operation which began at a valid address but extended beyond the last surface of the disc cartridge.
12	Unsafe	A malfunction exists in the selected drive.
13	Check Error	The checkword on the disc does not match the checkword calculated by the controller.
14	Data Late	The data channel failed to respond in time to a data channel request.
15	Error	One or more of the bits 10-14 is set to 1.

READ DISC ADDRESS

DIC <f> ac, DKP



The contents of the Disc Address Register and the Sector Counter are placed in bits 0-15 of the specified AC. After the data transfer takes place, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:



Bits	Name	Contents
0-1	Drive	Number of the drive selected.
2	Format	The selected drive is in the format mode. This mode should not be used.
3-6	---	Reserved for future use. If these bits were loaded with a SPECIFY DISC ADDRESS AND SECTOR COUNT instruction, they will be read into the specified AC when a READ DISC ADDRESS instruction is issued.
7	Surface	The surface number of the active head on the drive.
8-11	Sector	The sector number of the sector immediately following the last sector processed. If the last sector read or written was 13_8, this number will be 14_8 even though there is no such sector. If the operation is still in progress, this is the sector currently being read or written.
12-15	-Sector Count	The two's complement of the number of sectors left to be processed.

READ MEMORY ADDRESS COUNTER

DIB<f> ac,DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	1	F		0	1	1	0	1	1

The contents of the Memory Address Counter are placed in bits 1-15 of the specified AC. Bit 0 is set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. When the Memory Address Counter is read after a Write operation, the contents point to a memory location two greater than the location of the last word written on the disc cartridge. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	MEMORY ADDRESS														

Bits	Name	Contents
0	---	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used in a data channel transfer.

PROGRAMMING

The preparation of a moving-head disc pack for data channel transfers is divided into three distinct phases: I, selecting the drive unit, the surface, the sector and the number of sectors; II, positioning the Read/Write heads over the correct cylinder; and III, starting the Read or Write operation. The results of issuing commands in each phase should be checked for errors before proceeding to the next phase.

Phase I: Select the Drive, Surface, Sector and Number of Sectors

The initial selection of a disc drive is performed as follows: a SPECIFY DISC ADDRESS AND SECTOR COUNT instruction (DOC) is issued to the controller to select the drive unit, the surface of the cartridge, the first sector to be read or written, and the two's complement of the number of sectors to be transferred in the operation. The drive units are numbered 0-3; the surfaces are numbered 0 and 1; the sectors are numbered 0-13_8; the maximum number of sectors which can be transferred in one operation is sixteen. Care should be taken to insure that the parameters specified in this initial selection do not exceed the capacity of the disc cartridge.

Once the drive unit is chosen, the status of that drive must be checked to determine if the drive is ready to proceed. The status is checked by issuing a READ STATUS instruction (DIA) and examining the Drive Ready flag. If the Drive Ready flag is set to 1, the program can proceed to Phase II. If it is set to 0, the program should not issue any commands to that drive unit until it is in the ready state.

Phase II: Position the Heads

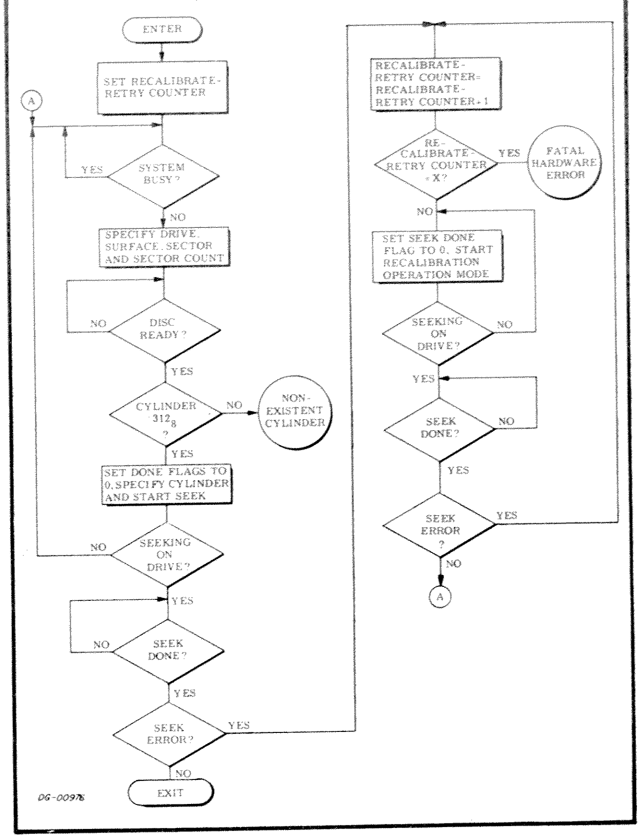
The heads are positioned over the desired cylinder as follows: a SPECIFY COMMAND AND CYLINDER instruction (DOA) is issued to the controller. This instruction should contain the number of the cylinder desired and the Seek command. The cylinders are numbered from 0-312_8. The instruction should also set both the Read/Write Done flag and the Seek Done flag for the selected drive to 0. The Seek operation is initiated by a Pulse command. While the drive is seeking, the Seeking On Drive flag for that drive is set to 1. When the heads have finished moving to the specified cylinder, the Seeking On Drive flag for the selected unit is set to 0 and the Seek Done flag for that drive unit is set to 1, thus initiating a program interrupt request.

The program should then check the Status Register to determine if a seek error has occurred as a result of a faulty Seek operation. If no errors have occurred, the program can proceed to Phase III.

The heads of the selected drive unit can be forced to cylinder 0 by the Recalibrate operation. A Recalibrate operation is performed as follows: a SPECIFY COMMAND AND CYLINDER instruction is issued to the controller. This instruction should contain the Recalibrate command and should also set both the Read/Write Done and the Seek Done flags for the selected drive unit to 0. The operation is initiated by a Pulse command. While the drive is being recalibrated, the Seeking On Drive flag for the selected drive is set to 1. Once the Recalibrate operation is completed, the Seeking On Drive flag is set to 0, the Seek Done flag for the selected drive is set to 1 and a program interrupt request is initiated.

When the program places a drive in the seek mode of operation, the controller is free to accept commands to the other drives under its direction. Therefore, once one or more drives are performing Seek operations, one of the other drives can perform a Read or a Write operation. If the program is simultaneously managing several drives with one controller, the error indicating flags in the Status Register apply only to the most recently selected drive unit, i. e., the unit specified in the last SPECIFY DISC ADDRESS AND SECTOR COUNT instruction issued.

INITIAL SELECTION AND HEAD POSITIONING

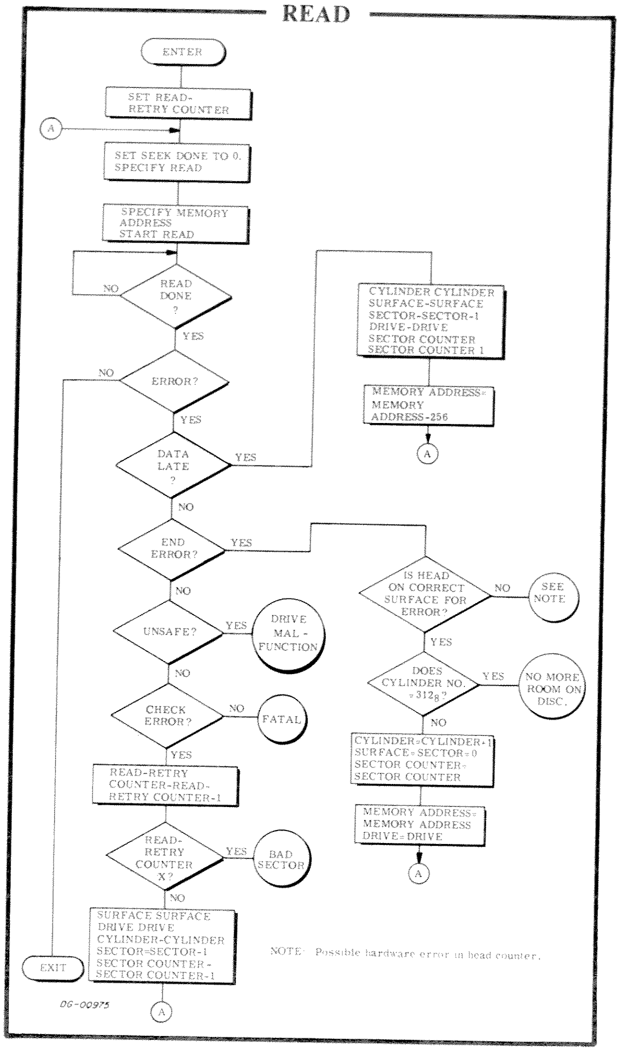

ENTER
SET RECALIBRATE-RETRY COUNTER
A
YES
SYSTEM BUSY?
NO
SPECIFY DRIVE, SURFACE, SECTOR AND SECTOR COUNT
NO
DISC READY?
YES
CYLINDER 312_8 ?
NO
NON-EXISTENT CYLINDER
YES
SET DONE FLAGS TO 0, SPECIFY CYLINDER AND START SEEK
NO
SEEKING ON DRIVE?
YES
NO
SEEK DONE?
YES
SEEK ERROR?
YES
NO
EXIT
DG-00976
RECALIBRATE-RETRY COUNTER= RECALIBRATE-RETRY COUNTER+1
RE-CALIBRATE-RETRY COUNTER =X?
YES
FATAL HARDWARE ERROR
NO
SET SEEK DONE FLAG TO 0, START RECALIBRATION OPERATION MODE
SEEKING ON DRIVE?
NO
YES
SEEK DONE?
NO
YES
SEEK ERROR ?
YES
NO
A

Phase III: Read or Write

A Read operation transfers blocks of data from the disc to the computer's memory, via the data channel. A block of data contains 256 words. Up to 16 blocks may be transferred in one Read operation. A Write operation transfers blocks of data from the computer's memory, via the data channel, to the disc. Again, up to 16 blocks of data may be transferred in one operation.

Read or Write operations are performed in a series of steps which are virtually identical. The parameters of the data transfer must be specified and finally the operation is initiated.

Read


READ
ENTER
SET READ-RETRY COUNTER
A
SET SEEK DONE TO 0. SPECIFY READ
SPECIFY MEMORY ADDRESS START READ
READ DONE ?
NO
YES
ERROR?
NO
YES
DATA LATE ?
YES
NO
CYLINDER CYLINDER SURFACE-SURFACE SECTOR-SECTOR-1 DRIVE-DRIVE SECTOR COUNTER SECTOR COUNTER 1
MEMORY ADDRESS= MEMORY ADDRESS-256
A
END ERROR?
YES
NO
IS HEAD ON CORRECT SURFACE FOR ERROR?
NO
SEE NOTE
YES
UNSAFE?
YES
DRIVE MAL-FUNCTION
NO
DOES CYLINDER NO. =312$_8$?
YES
NO MORE ROOM ON DISC.
NO
CHECK ERROR?
NO
FATAL
YES
CYLINDER=CYLINDER+1 SURFACE=SECTOR=0 SECTOR COUNTER= SECTOR COUNTER
READ-RETRY COUNTER-READ-RETRY COUNTER-1
MEMORY ADDRESS= MEMORY ADDRESS DRIVE=DRIVE
A
READ-RETRY COUNTER X?
YES
BAD SECTOR
NO
SURFACE SURFACE DRIVE DRIVE CYLINDER-CYLINDER SECTOR=SECTOR-1 SECTOR COUNTER-SECTOR COUNTER-1
EXIT
A
NOTE: Possible hardware error in head counter.
DG-00975

A Read operation is performed as follows: the storage location in memory for the first word to be read from the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be read and the starting sector were specified in Phase I.

The Read command is then loaded into the Command Register with a SPECIFY COMMAND AND CYLINDER instruction (DOA). It is not necessary to load the cylinder number again, but it is good practice since other disc subsystems may require this information. The Read operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

Once the Read operation is begun, the drive unit waits until the desired sector passes under the head and then starts reading the sequential bits of the first word in the sector. When a word is fully assembled, the controller transfers the word to the computer's memory via the data channel. Each time a word is transferred to memory, the Memory Address Counter is automatically incremented.

When the 256 words from the sector have been read, and the checkword at the end of the sector verified, the sector counter is incremented by one.

If the sector counter does not overflow, the next sector is read. This process continues until either the sector counter overflows or the last sector on the surface is read. In the case where the last sector on surface 0 is read, the sector counter has not overflowed, the drive will automatically continue the operation by reading the first sector on surface 1 in the same cylinder.

The Read operation then continues until the sector counter indicates, by overflowing, that the specified number of sectors have been read. Upon completion of the Read operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Write

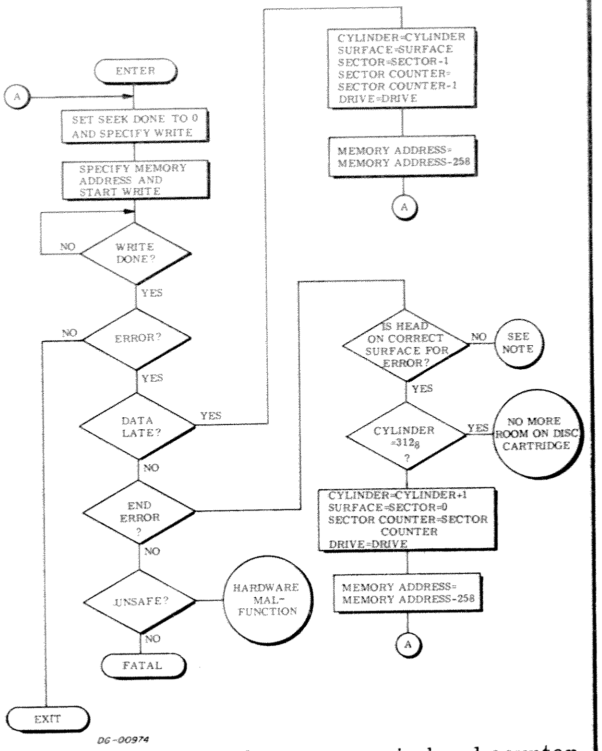
A Write operation is performed as follows: the storage location in memory of the first word to be written on the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be written and the number of the starting sector were specified in Phase I. The Write command is then loaded into the Command Register with a SPECIFY COMMAND AND CYLINDER instruction (DOA). It is not necessary to load the cylinder number again, but it is good practice since other disc drives may require this information. The Write operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

Once the Write operation is initiated, the controller reads two words from the computer's memory, via the data channel and then waits for the desired sector to pass under the head. Each time the controller reads a word from the computer's memory, the Memory Address Counter is incremented. Once the desired sector is encountered, the bits of each word are sequentially written. When the 256 words in the sector have been written, the controller writes the checkword it calculated from the data during the Write operation. The sector counter is then incremented by one.

If the sector counter does not overflow, the next sector is written. This process continues until either the sector counter overflows or the last sector on the surface is written. In the case when the last sector on surface 0 is written, and the sector counter has not overflowed, the drive will automatically continue the operation by writing the first sector on surface 1 in the same cylinder.

The Write operation then continues until the sector counter indicates, by overflowing, that the specified number of sectors have been written. Upon completion of the Write operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt is initiated.

WRITE


ENTER
A
SET SEEK DONE TO 0 AND SPECIFY WRITE
SPECIFY MEMORY ADDRESS AND START WRITE
WRITE DONE?
NO
YES
ERROR?
NO
YES
DATA LATE?
YES
NO
END ERROR ?
NO
UNSAFE?
HARDWARE MAL-FUNCTION
NO
FATAL
EXIT
CYLINDER=CYLINDER
SURFACE=SURFACE
SECTOR=SECTOR-1
SECTOR COUNTER=SECTOR COUNTER-1
DRIVE=DRIVE
MEMORY ADDRESS=MEMORY ADDRESS-258
A
IS HEAD ON CORRECT SURFACE FOR ERROR?
NO
SEE NOTE
YES
CYLINDER =312$_8$?
YES
NO MORE ROOM ON DISC CARTRIDGE
CYLINDER=CYLINDER+1
SURFACE=SECTOR=0
SECTOR COUNTER=SECTOR COUNTER
DRIVE=DRIVE
MEMORY ADDRESS=MEMORY ADDRESS-258
A
DG-00974

Note: Possible hardware error in head counter.

TIMING

The disc cartridge rotates at a speed of 1500 rpm; a complete revolution requires 40 milliseconds. A register, containing the identification number of the sector currently passing under the head, is used to reduce the sector accessing time. This feature allows the subsystem to carry out a Read or a Write operation the first time the desired sector passes under the head without waiting for a track address check.

Since each data record in a sector is preceded by a 0.5 millisecond gap, used for synchronization, the minimum sector access time is 0.5 milliseconds. The maximum is 40.5 milliseconds so the average is 20.5 milliseconds.

The time required to position the heads (the seek time) is dependent on the number of cylinders the heads must move past in a Seek operation. A maximum of 15 milliseconds is required to move the heads from one cylinder to the adjacent cylinder. The time required to move from cylinder 0 to cylinder 312_8, or vice versa, is 135 milliseconds, maximum.

Any time a Seek or Recalibrate operation is initiated by a Pulse command, the controller requires 50 microseconds to respond to the command. Therefore, the program must wait 50 microseconds after initiating a head movement operation before any other disc command can be issued.

A sector passes under the head in 3.33 milliseconds while the data block in the sector passes under the head in 2.84 milliseconds. Since there are 256 data words in a sector, a data channel request occurs every 11.1 microseconds. This corresponds to a data transfer rate of 90,000 words/second. Since the controller is doubled buffered, the maximum allowable data channel latency is 22.2 microseconds. If the data channel does not respond within this time, both the Data Late and the Error flags are set to 1. Once this error occurs, the processing of the current sector is completed and the command is terminated, even if the operation was scheduled to transfer additional sectors. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

ERROR CONDITIONS

During Initial Selection

If the program specifies a non-existent sector ($>13_8$) no indication of this error is given. If the subsystem then attempts a Read or a Write operation, the drive unit will search forever for that non-existent sector. The subsystem can be released from this search by having the program issue an I/O RESET instruction (IORST).

During Head Positioning

If the program issues a SPECIFY COMMAND AND CYLINDER instruction which specifies a non-existent cylinder ($>312_8$) and then places the drive unit in the seek mode, the Seek operation is terminated. Both the Seek Error and the Seek Done flags for that unit are set to 1. When the Seek Done flag is set to 1, a program interrupt request is initiated.

If any Seek operation to a valid cylinder number results in a Seek Error, the drive unit should be recalibrated.

During Reading

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Read operation is initiated.

An error can occur when the subsystem is reading a series of sectors in one operation if the Read operation exceeds the number of sectors available. When the last sector on surface 1 is read and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

If the checkword read at the end of a sector differs from that calculated by the controller, both the Error and the Check Error flags are set to 1. The Read operation is terminated, even if more sectors were supposed to be read; the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Check Error indicates that at least one word in the last sector read contains an error.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The reading of the current sector continues, but once that sector has been read, the Read operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word from the last sector read was not correctly transferred to memory.

During Writing

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Write operation is initiated.

An error can occur when the subsystem is writing a series of sectors in one operation if the Write operation exceeds the number of sectors available. When the last sector on surface 1 is written and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to sector 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The writing of the current sector continues, but once that sector has been written, the Write operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word in the sector was not written properly.

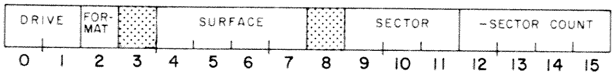
THE 4048A DISC PACK SUBSYSTEM

SUMMARY

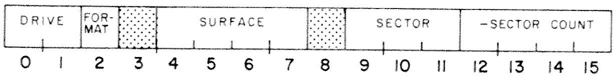
MNEMONIC (FIRST CONTROLLER)	DKP
DEVICE CODE (FIRST CONTROLLER)	33_8
MNEMONIC (SECOND CONTROLLER)	DKP1
DEVICE CODE (SECOND CONTROLLER)	73_8
PRIORITY MASK BIT	7
SURFACES/UNIT	10
TRACKS/SURFACE (CYLINDERS)	203
SECTORS/TRACK	6
WORDS/SECTOR	256
TOTAL STORAGE CAPACITY (WORDS)	3,118,080
MAXIMUM TRANSFER RATE (WORDS/SEC)	78,000
ALLOWABLE DATA CHANNEL LATENCY (μSEC)	25.6
SEEK TIME MAX/MIN (mSEC)	60/10
SECTOR ACCESS TIME MAX/MIN (mSEC)	46.3/0

ACCUMULATOR FORMATS

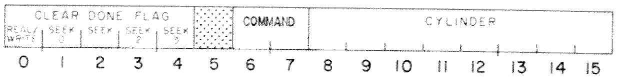
SPECIFY DISC ADDRESS AND SECTOR COUNT .... (DOC)



READ DISC ADDRESS .... (DIC)



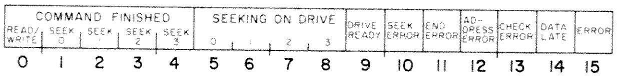
SPECIFY COMMAND AND CYLINDER... (DOA)



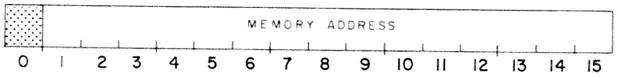
COMMANDS

00	Read	10	Seek
01	Write	11	Recalibrate

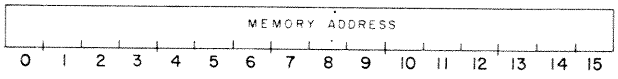
READ STATUS ....(DIA)



LOAD MEMORY ADDRESS COUNTER... (DOB)



READ MEMORY ADDRESS COUNTER.... (DIB)



S, C AND P FUNCTIONS

S	Set Busy to one, Done to zero and start a Read or a Write operation.
C	Set Busy to zero, Done to zero and stop all operations.
P	Start a Seek or a Recalibrate operation.

INTRODUCTION

The 4048A disc pack subsystem utilizes a removable disc pack containing 10 (0-11_8) program accessible surfaces. There are 203 (0-312_8) cylinders on the disc pack. Each of the preformatted tracks in a cylinder contains 6 (0-5) sectors, each of which stores 256 (400_8) 16-bit words and contains a checkword. The data storage capacity is 1,536 words/track, 15,360 words/cylinder or 3,118,080 words/pack. Words are transferred to and from the subsystem via the data channel at a rate of 78,000 words per second. Up to 16 sectors containing 4096 (10000_8) words can be transferred in one operation.

The controller for the subsystem, when coupled to the adapter, can direct the activities of up to four drive units. Any number of these units can be performing Seek operations simultaneously, but only one drive can be reading or writing at any one time.

INSTRUCTIONS

The disc drive controller contains four program accessible registers: a 15-bit Memory Address Counter, a 16-bit Status Register, a 16-bit combined Command/Cylinder Select Register and a combined disc Address/Sector Counter Register. The Memory Address Counter is self-incrementing and contains the memory location of the next 16-bit word to be either read from or written on the disc. The Status Register contains all the information flags for the controller and the selected drive as well as 8 flags which indicate when any drive completes a Seek or a Recalibrate operation. Any of the 4 Seek Done flags as well as the Read/Write Done flag are able to initiate a program interrupt request when they are set to 1. The combined Command/Cylinder Select Register contains the command last issued to the subsystem and the number of the desired cylinder on the disc surface. The combined Disc Address/Sector Counter contains the surface and sector location of the active head and the two's complement of the number of sectors to be either read from or written on the disc. The Sector Counter is self-incrementing after each sector is read or written.

Six instructions are used to program data channel transfers to and from the disc pack. Three of these instructions are used to supply all of the necessary data to the controller for any disc operation. The remaining three instructions allow the program to determine, in detail, the current state of the disc pack subsystem.

The disc controller's Busy and Done flags are controlled using all three of the device flag commands as follows:

f = S Set the Busy flag to 1, the Done and all error indicating flags to 0, and initiate a Read or a Write operation, depending on the contents of the Command Register. The error indicating flags are: Seek Error, End Error, Address Error or Unsafe, Check Error, Data Late, and Error.

f = C Set the Busy, Done and all error indicating flags to 0 and stop all positioning and data transferring operations.

f = P Initiate either a Seek or a Recalibrate operation, depending on the contents of the Command Register.

SPECIFY DISC ADDRESS AND SECTOR COUNT

DOC f ac, DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	0	F		0	1	1	0	1	1

Bits 0-15 of the specified AC are loaded into the controller's combined Address Register/Sector Counter. Bits 3 and 8 are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DRIVE		FOR	▨	SURFACE				▨	SECTOR			-SECTOR COUNT			

Bits	Name	Function
0-1	Drive	Select the drive 0, 1, 2, or 3.
2	Format	If 1, place the drive in the format mode.
3	----	Reserved for future use.
4-7	Surface	Select the surface (head), $0-11_8$, for the start of a Read or Write operation.
8	----	Reserved for future use.
9-11	Sector	Select the starting sector, 0-5, for the start of a Read or a Write operation.
12-15	-Sector Count	Specify the two's complement of the number of sectors to be read or written in one operation (maximum of 16).

READ STATUS

DIA<f> ac, DKP

0	1	1	AC		0	0	1	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Status Register are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

COMMAND FINISHED					SEEKING ON DRIVE										
READ/ WRITE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	0	1	2	3	DRIVE READY	SEEK ERROR	END ERROR	ADDR. ERROR	CHECK ERROR	DATA LATE	ERROR
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Meaning When 1
0	Read/Write Done	The subsystem has completed a Read or Write operation.
1-4	Seek Done	Drive 0-3, respectively, has completed a Seek or Recalibrate operation. More than one of these bits can be set at any time.
5-8	Seeking On Drive	Drive unit 0-3, respectively, is currently performing a Seek or Recalibrate operation.
9	Disc Ready	The selected drive is ready to carry out a command.
10	Seek Error	The selected drive did not carry out the Seek or Recalibrate operation which was initiated.
11	End Error	The selected drive attempted to continue a Read or Write operation which began at a valid address but extended beyond the last surface in the disc pack.
12	Address Error or Unsafe	Either the address read at the beginning of the track did not match the specified address or a malfunction exists in the selected drive.
13	Check Error	The checkword read from the disc at the end of a sector does not match the checkword calculated by the controller.
14	Data Late	The data channel failed to respond in time to a data channel request.
15	Error	One or more of the bits 10-14 is 1.

SPECIFY COMMAND AND CYLINDER

DOA<f> ac, DKP

0	1	1	AC		0	1	0	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-15 of the specified AC are loaded into the controller's combined Command/Cylinder Select Register. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

CLEAR DONE FLAG															
READ/ WRITE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	▨	COMMAND		CYLINDER							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0	Clear Read/ Write Done	Set the controller's Done flag to 0; set the following error indicating flags to 0: Seek Error, Address Error or Unsafe, End Error, and Check Error.
1-4	Clear Seek Done	Set the Seek Done flags to 0 for the drives 0-3, respectively.
5	----	Reserved for future use.
6-7	Command	Specify the command for the selected drive as follows: 00 Read with a Start command. 01 Write with a Start command. 10 Seek with a Pulse command to the cylinder specified in bits 8-15 of this accumulator. 11 Recalibrate with a Pulse command.
8-15	Cylinder	Specify the cylinder 0-312_8, for a Seek, Read or Write operation.

LOAD MEMORY ADDRESS COUNTER

DOB<f> ac, DKP

0	1	1	AC		1	0	0	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-15 of the specified AC are loaded into the controller's Memory Address Counter. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

▨	MEMORY ADDRESS														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0	---	Reserved for future use.
1-15	Memory	Location of the next word in memory to be used for a data channel transfer. Bit 0 is loaded and read back but it does not affect the data transfer.

READ DISC ADDRESS

DIC <f> ac, DKP

0	1	1	AC		1	0	1	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the disc address register and the Sector Counter are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

DRIVE		FOR	▨	SURFACE				▨	SECTOR			-SECTOR COUNT			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-1	Drive	Number of the selected drive.
2	Format	The selected drive is in the format mode.
3-7	Surface	The surface number of the active head on the drive. Although bit 3 specifies a surface which does not exist, if it were loaded by a DOC instruction, it will be set when read back.
8-11	Sector	The number of the sector immediately following the last sector read or written. Although bit 8 specifies a sector which does not exist, if it were loaded by a DOC instruction, it will be set when read back.
12-16	-Sector Count	The two's complement of the number of sectors left to be read or written.

READ MEMORY ADDRESS COUNTER

DIB<f> ac, DKP

0	1	1	AC		0	1	1	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Memory Address Counter are placed in bits 0-15 of the specified AC. After the data transfer, the controll's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

MEMORY ADDRESS															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-15	Memory Address	Location of the next word in memory to be used for a data channel transfer. If bit 0 was 0 in a LOAD MEMORY COUNTER instruction and is a 1 when read, the data transfer exceeded the capacity of a 32K machine. The excess words transferred are in low core.

PROGRAMMING

The preparation of a moving head disc pack for data channel transfers is divided into three distinct phases: I, selecting the drive, the surface and the sector; II, positioning the heads over the correct cylinder; and III, starting the Read or the Write operation. The results of issuing commands in each phase should be checked for errors before proceeding to the next phase.

Phase I: Select the Drive, Surface, Sector and Number of Sectors

The initial selection of a disc drive is performed as follows: a SPECIFY DISC ADDRESS AND SECTOR COUNT instruction (DOC) is issued to the controller to select the drive, the surface of the disc pack, the first sector to be read or written, and the two's complement of the number of sectors to be transferred in the operation. The drives are numbered 0-3; the surfaces are numbered 0-11_8; the sectors are numbered 0-5_8; the maximum number of sectors which can be transferred in one operation is 16. Care should be taken to insure that the parameters specified in this initial selection do not exceed the capacity of the disc pack.

Once the drive unit is chosen, the status of that drive must be checked to determine if the drive is ready to proceed. The status is checked by issuing a READ STATUS instruction (DIA) and examining the Ready flag. If the Ready flag is set to 1, the program can proceed to Phase II. If it is set to 0, the program should not issue any commands to that drive unit until it is in the ready state.

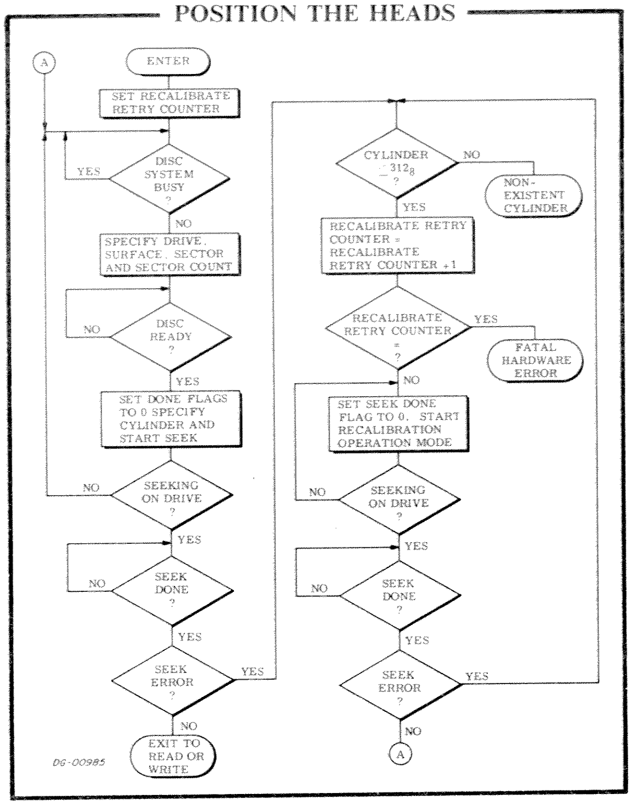
Phase II: Position the Heads

The heads are positioned over the desired cylinder as follows: a SPECIFY COMMAND AND CYLINDER instruction (DOA) is issued to the controller. This instruction should contain the number of the cylinder desired and the Seek command. The cylinders are numbered from $0\text{-}312_8$. The instruction should also set both the Read/Write Done flag and the Seek Done flag for the selected drive to 0. The Seek operation is initiated by a Pulse command. While the drive is seeking, the Unit Ready flag for that drive is set to 0 and the Seeking On Drive flag for the selected drive is 1. When the heads have finished moving to the specified cylinder, the Seeking On Drive flag is set to 0, and both the Unit Ready and the Seek Done flags for that drive unit are set to 1, thus initiating a program interrupt request.

The program should then check the Status Register to determine if a seek error has occurred as a result of a faulty Seek operation. If no errors have occurred, the program can proceed to Phase III.

The heads of the selected drive unit can be forced to cylinder 0 by the Recalibrate operation. A Recalibrate operation is performed as follows: a SPECIFY COMMAND AND CYLINDER instruction is issued to the controller. This instruction should contain the Recalibrate command and should also set both the Read/Write Done and the Seek Done flags for the selected drive unit to 0. The operation is initiated by a Pulse command. While the drive is being recalibrated, the Unit Ready flag for the selected drive is set to 0 and the Seeking On Drive flag for the selected drive is set to 1. Once the Recalibrate operation is completed, the Seeking On Drive flag is set to 0, the Unit Ready flag is set to 1, the Seek Done flag for the selected drive is set to 1 and a program interrupt request is initiated.

POSITION THE HEADS


A
ENTER
SET RECALIBRATE RETRY COUNTER
DISC SYSTEM BUSY ?
YES
NO
SPECIFY DRIVE, SURFACE, SECTOR AND SECTOR COUNT
DISC READY ?
NO
YES
SET DONE FLAGS TO 0 SPECIFY CYLINDER AND START SEEK
SEEKING ON DRIVE ?
NO
YES
SEEK DONE ?
NO
YES
SEEK ERROR ?
YES
NO
EXIT TO READ OR WRITE
DG-00985
CYLINDER ≤ 312₈ ?
NO
NON-EXISTENT CYLINDER
YES
RECALIBRATE RETRY COUNTER = RECALIBRATE RETRY COUNTER +1
RECALIBRATE RETRY COUNTER = ?
YES
FATAL HARDWARE ERROR
NO
SET SEEK DONE FLAG TO 0. START RECALIBRATION OPERATION MODE
SEEKING ON DRIVE ?
NO
YES
SEEK DONE ?
NO
YES
SEEK ERROR ?
YES
NO
A

When the program places a drive in the seek mode of operation, the controller is free to accept commands to the other drives under its direction. Therefore, once one or more drives are performing Seek operations, one of the other drives can perform a Read or a Write operation. If the program is simultaneously managing several drives with one controller, the error indicating flags in the Status Register apply only to the most recently selected drive unit, i. e., the unit specified in the last SPECIFY DISC ADDRESS AND SECTOR COUNT instruction issued.

Phase III: Read or Write

A Read operation transfers blocks of data stored in the disc subsystem to the computer, via the data channel. A block of data contains 256 16-bit words. Up to 16 blocks can be transferred in one Read operation. A Write operation transfers blocks of data from the computer's memory via the data channel, and stores the data in the disc subsystem. Again, up to 16 blocks of data may be transferred in one operation. Read or Write operations are performed in a series of steps which are virtually identical. The parameters of the data transfer must be specified and finally the operation is initiated.

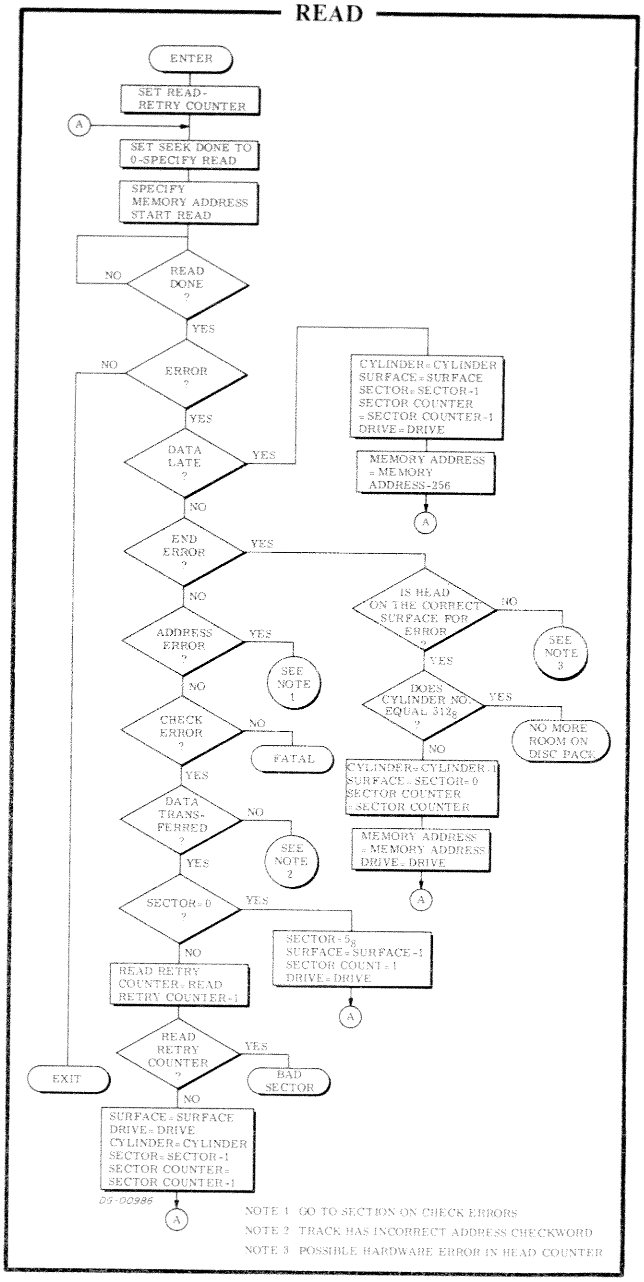
Read

A Read operation is performed as follows: the storage location in memory for the first word to be read from the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be read and the starting sector were specified in Phase I.

The Read command is then loaded into the Command Register with a SPECIFY COMMAND CYLINDER instruction (DOA). It is not necessary to load the cylinder number again, but it is good practice since other disc drives may require this information. The Read operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

Once the Read operation is initiated, the drive waits until the address field at the beginning of the track passes under the head and performs an address check. If the address read is correct, the controller waits until the specified sector is encountered. The drive then starts reading the sequential bits of the first word of the sector. When a word is fully assembled, the controller transfers the word to the computer's memory via the data channel. Each time a word is transferred to memory, the Memory Address Counter is automatically incremented.

When the 256 words from the sector have been read, and the checkword at the end of the sector verified, the sector counter is automatically incremented by one.



NOTE 1 GO TO SECTION ON CHECK ERRORS
NOTE 2 TRACK HAS INCORRECT ADDRESS CHECKWORD
NOTE 3 POSSIBLE HARDWARE ERROR IN HEAD COUNTER

If the sector counter does not overflow, the next sector is read and this process continues until either the sector counter overflows or the last sector on the surface is read. In the case where the last sector on surface $0\text{-}10_8$ is read, and the sector counter has not overflowed, the drive will automatically continue the operation by reading the first sector on the next surface in the same cylinder.

The Read operation then continues until the sector counter indicates, by overflowing, that the specified number of sectors have been read. Upon completion of the Read operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Write

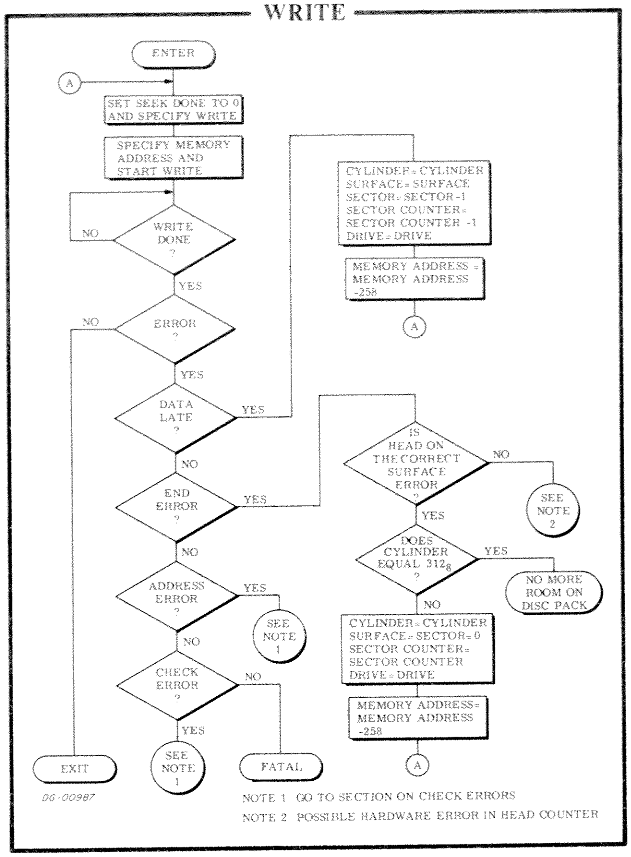
A Write operation is performed as follows: the storage location in memory of the first word to be written on the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be written and the number of the starting sector were specified in Phase I.

The Write command is then loaded into the Command Register with a SPECIFY COMMAND AND CYLINDER instruction (DOA). It is not necessary to load the cylinder number again, but it is good practice since other disc drives may require this information. The Write operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

Once the Write operation is initiated, the controller reads two words from the computer's memory via the data channel and then waits for the address field at the beginning of the track to pass under the head. Each time the controller reads a word from the computer's memory, the Memory Address Counter is automatically incremented. Once the address field is encountered, the address check is performed automatically. If the address read from the disc is correct, the controller waits until the specified sector is encountered. The bits of each word then are sequentially written. When the 256 words in the sector have been written, the controller writes the checkword it calculated from the data it wrote on the disc. The sector counter is automatically incremented by one.

If the sector counter does not overflow, the next sector is written. This process continues until either the sector counter overflows of the last sector on the surface is written. In the case when the last sector on surface $0\text{-}10_8$ is written, and the sector counter has not overflowed, the drive will automatically continue the operation by writing the first sector on the next surface in the same cylinder.

The Write operation then continues until the sector counter indicates, by overflowing, that the specified number of sectors have been written. Upon completion of the Write operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt is initiated.


WRITE
ENTER
A
SET SEEK DONE TO 0 AND SPECIFY WRITE
SPECIFY MEMORY ADDRESS AND START WRITE
WRITE DONE ?
NO
YES
ERROR ?
NO
YES
DATA LATE ?
YES
NO
END ERROR ?
YES
NO
ADDRESS ERROR ?
YES
SEE NOTE 1
NO
CHECK ERROR ?
NO
YES
EXIT
SEE NOTE 1
FATAL
CYLINDER = CYLINDER
SURFACE = SURFACE
SECTOR = SECTOR -1
SECTOR COUNTER = SECTOR COUNTER -1
DRIVE = DRIVE
MEMORY ADDRESS = MEMORY ADDRESS -258
A
IS HEAD ON THE CORRECT SURFACE ERROR ?
NO
SEE NOTE 2
YES
DOES CYLINDER EQUAL 312_8 ?
YES
NO MORE ROOM ON DISC PACK
NO
CYLINDER = CYLINDER
SURFACE = SECTOR = 0
SECTOR COUNTER = SECTOR COUNTER
DRIVE = DRIVE
MEMORY ADDRESS = MEMORY ADDRESS -258
A

DG-00987

NOTE 1 GO TO SECTION ON CHECK ERRORS

NOTE 2 POSSIBLE HARDWARE ERROR IN HEAD COUNTER

Format

A new disc pack is unformatted so a Disc Pack Formatter program is available to initialize the disc pack with the proper prerequisites for reading from or writing on the disc pack.

This program is the 4048 Disc Pack Formatter Program #096-000039 (tape #095-000072).

A disc pack may also be reformatted when repeated recalibrations of the heads do not eliminate an Address Error occurring during a Read or a Write operation. All data is lost when a disc pack is reformatted.

TIMING

The disc pack rotates at a speed of 2400rpm; a complete revolution requires 25 milliseconds. Since the controller must wait until the index point on the disc passes the head before the search for the desired sector can be started, the latency time before starting the sector search can be as much as 25 milliseconds (average of 12.5 milliseconds. Once the search for the correct sector begins, the time taken to reach that sector depends on which sector is specified: sector 0 is reached almost immediately while sector 5 requires 21.3 milliseconds before reading or writing can commence.

The total sector access time is therefore the time required to reach the address field plus the time required to reach the desired sector. The maximum is 46.3 milliseconds (25 to reach the address field plus the 21.3 milliseconds to reach sector 5).

The time required to position the heads (the seek time) is dependent on the number of cylinders the heads must move past in a Seek operation. A maximum of 10 milliseconds is required to move the heads from one cylinder to the adjacent cylinder. The time required to move from cylinder 0 to cylinder 312_8, or vice versa, is 60 milliseconds, maximum. A Recalibrate operation requires a maximum of 300 milliseconds.

If a Seek or a Recalibrate operation is not completed within 1 second, the Seek Error and Error flags are set to 1. The Done flag is set to 1 and a program interrupt request is initiated.

A sector passes under the head in 4.17 milliseconds while the data block in the sector passes under the head in 3.28 milliseconds. Since there are 256 data words in a sector, a data channel request occurs every 12.8 microseconds. This corresponds to a data transfer rate of 78,000 words/second. Since the controller is double buffered, the maximum allowable data channel latency is 25.6 microseconds. If the data channel does not respond within this time, both the Data Late and the Error flags are set to 1. Once this error occurs, the processing of the current sector is completed and the command is terminated, even if the operation was scheduled to transfer additional sectors. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

For the greatest efficiency when processing an entire cylinder, the program should be coded to process two tracks at a time since the head must wait for the index point on the surface to pass the head before each command can begin. However, after completing the processing of these two tracks, enough time is available (.5msec) to prepare the disc pack to read or write the next track without waiting for a full rotation of the disc to correctly position sector zero. This procedure allows the head to start processing immediately after a command is issued without having to wait for the drive to search for the correct sector before beginning the operation.

ERROR CONDITIONS

During Initial Selection

If the program specifies a non-existent sector ($>5_8$) no indication of this error is given. If the subsystem then attempts a Read or a Write operation, the drive unit will search forever for that non-existent sector. The subsystem can be released from this search by having the program issue an I/O RESET instruction (IORST).

During Head Positioning

If the program issues a SPECIFY COMMAND AND CYLINDER instruction which specifies a non-existent cylinder ($>312_8$) and then places the drive unit in the seek mode, the Seek operation is terminated. Both the Seek Error and the Seek Done flags for that unit are set to 1. When the Seek Done flag is set to 1, a program interrupt request is initiated.

If any Seek operation to a valid cylinder number results in a Seek Error, the drive unit should be recalibrated

During Reading

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Read operation is initiated.

An error can occur when the subsystem is reading a series of sectors in one operation if the Read operation exceeds the number of sectors available. When the last sector on surface 11_8 is read and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The reading of the current sector continues, but once that sector has been read, the Read operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word from the sector read was not correctly transferred to memory.

During Writing

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Write operation is initiated.

An error can occur when the subsystem is writing a series of sectors in one operation if the Write operation exceeds the number of sectors available. When the last sector on surface 11_8 is written and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to sector 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The writing of the current sector continues, but once that sector has been written, the Write operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word in the sector was not written properly.

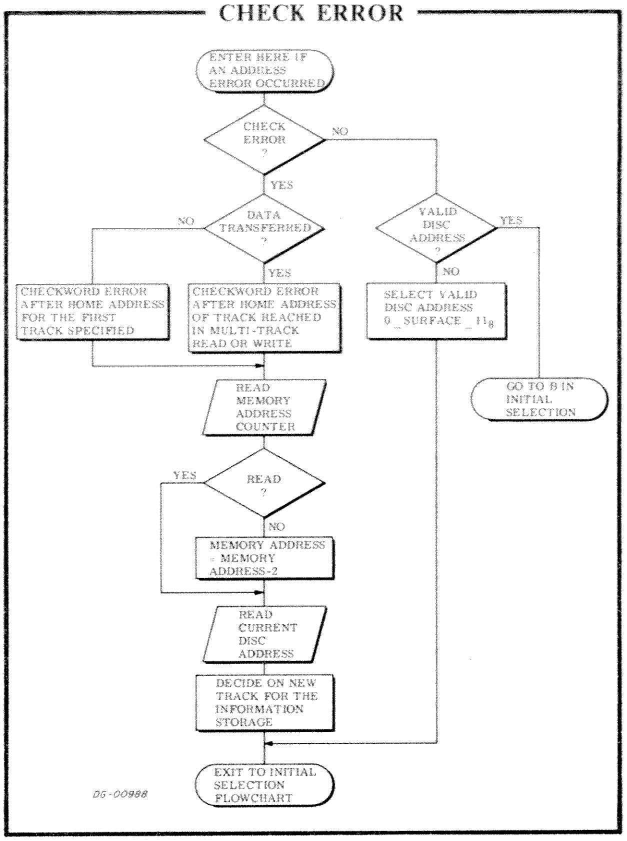
Check Error

A Check Error flag is set to 1 during a Read operation if the 16-bit checkword read after the data in the sector being processed does not match the checkword calculated by the controller during the Read operation.

The Check Error flag can also be set to 1 if a checkword error is detected when verifying the home address of a track in either a Read or a Write operation. Usually the Address Error flag is also set to 1 since an error has been made in reading the track address information. However, this Checkword Error can also occur without an accompanying Address Error when the checkword read at the end of the track home address is incorrect. In both of these cases, access to all the sectors on that track is denied.

Since the Check Error flag can be set to 1 without an Address Error flag under two circumstances in a Read operation, the programmer should verify which condition caused the flag to be set to 1. If the Check Error flag is posted and the Memory Address Counter has not changed, the checkword following the track home address is incorrect. In addition, if the Read operation does not extend to the next surface and data has been transferred, then the error is in the last sector processed.

CHECK ERROR


ENTER HERE IF AN ADDRESS ERROR OCCURRED
CHECK ERROR ?
NO
YES
DATA TRANSFERRED ?
NO
YES
CHECKWORD ERROR AFTER HOME ADDRESS FOR THE FIRST TRACK SPECIFIED
CHECKWORD ERROR AFTER HOME ADDRESS OF TRACK REACHED IN MULTI-TRACK READ OR WRITE
VALID DISC ADDRESS ?
YES
NO
SELECT VALID DISC ADDRESS 0 _ SURFACE _ 11_8
GO TO B IN INITIAL SELECTION
READ MEMORY ADDRESS COUNTER
READ ?
YES
NO
MEMORY ADDRESS = MEMORY ADDRESS-2
READ CURRENT DISC ADDRESS
DECIDE ON NEW TRACK FOR THE INFORMATION STORAGE
EXIT TO INITIAL SELECTION FLOWCHART
DG-00988

However, when a Read operation extends beyond one surface, and a Check Error occurs after the last sector on the first surface is read and before the first sector on the second surface is read, the error could be in either the last sector on the first surface or in the checkword for the home address of the track in the second surface. In each of these two cases, the Memory Address Counter and the Disc Address Register will contain identical information.

These two cases can be differentiated by performing a Read operation on the last sector of the first surface. If this Read is successful, the error is in the checkword of the home address of the track on the second surface. If this is the case, then that track on the second surface is inaccessible.

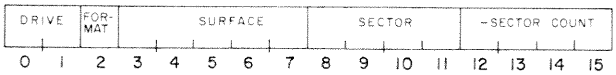
THE 4057A DISC PACK SUBSYSTEM

SUMMARY

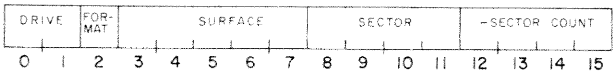
MNEMONIC (FIRST CONTROLLER)	DKP
DEVICE CODE (FIRST CONTROLLER)	33_8
MNEMONIC (SECOND CONTROLLER)	DKP1
DEVICE CODE (SECOND CONTROLLER)	73_8
PRIORITY MASK BIT	7
SURFACES/UNIT	20
TRACKS/SURFACE (CYLINDERS)	203
SECTORS/TRACK	12
WORDS/SECTOR	256
TOTAL STORAGE CAPACITY (WORDS)	12,472,320
MAXIMUM TRANSFER RATE (WORDS/SEC)	156,000
ALLOWABLE DATA CHANNEL LATENCY (μSEC)	12.8
SEEK TIME MAX/MIN (mSEC)	60/10
SECTOR ACCESS TIME MAX/MIN (mSEC)	48.2/0.2

ACCUMULATOR FORMATS

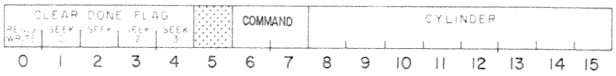
SPECIFY DISC ADDRESS SECTOR COUNT .... (DOC)



READ DISC ADDRESS .... (DIC)



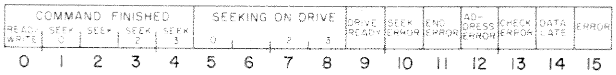
SPECIFY COMMAND AND CYLINDER ... (DOA)



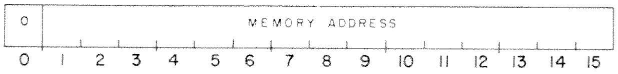
COMMANDS

00 Read	10 Seek
01 Write	11 Recalibrate

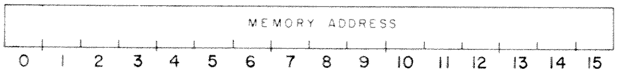
READ STATUS .... (DIA)



LOAD MEMORY ADDRESS COUNTER... (DOB)



READ MEMORY ADDRESS COUNTER....(DIB)



S, C AND P FUNCTIONS

S	Set Busy to one, Done to zero and start a Read or a Write operation.
C	Set Busy to zero, Done to zero and stop all operation.
P	Start a Seek or a Recalibrate operation.

INTRODUCTION

The 4057A disc pack subsystem utilizes a removable disc pack containing 20 ($0-23_8$) program accessible surfaces. There are 203 ($0-312_8$) cylinders on the disc pack. Each of the preformatted 20 tracks in a cylinder contains 12 ($0-13_8$) sectors, each of which stores 256 (400_8) 16-bit words and contains a checkword. The data storage capacity is 3072 words/track, 61,440 words/cylinder or 12,472,320 words/pack. Words are transferred to and from the subsystem via the data channel at a rate of 156,000 words per second. Up to 16 sectors containing 4096 ($10,000_8$) words can be transferred in one operation.

The controller for the subsystem, when coupled to the adapter, can direct the activities of up to four drive units. Any number of these units can be performing Seek operations simultaneously, but only one drive can be reading or writing at any one time.

INSTRUCTIONS

The disc drive controller contains four program accessible registers: a 15-bit Memory Address Counter, a 16-bit Status Register, a 16-bit combined Command/Cylinder Select Register and a combined disc Address/Sector Counter Register. The Memory Address Counter is self-incrementing and contains the memory location of the next 16-bit word to be either read from or written on the disc. The Status Register contains all the information flags for the controller and the selected drive as well as 8 flags which indicate the Seek status of all the drives in the subsystem. Any of the 4 Seek Done flags as well as the Read/Write Done flags are able to initiate a program interrupt request when they are set to 1. The combined Command/Cylinder Select Register contains the command last issued to the subsystem and the number of the desired cylinder on the disc surface. The combined Disc Address/Sector Counter contains the surface and sector location of the active head and the two's complement of the number of sectors to be either read from or written on the disc. The Sector Counter is self-incrementing after each sector is read or written.

Six instructions are used to program data channel transfers to and from the disc pack. Three of these instructions are used to supply all of the necessary data to the controller for any disc operation. The remaining three instructions allow the program to determine, in detail, the current state of the disc pack subsystem.

The disc controller's Busy and Done flags are controlled using all three of the device flag commands as follows:

f=S Set the Busy flag to 1, the Done and all error indicating flags to 0, and initiate a Read or a Write operation, depending on the contents of the command register. The error indicating flags are: Seek Error, End Error, Address Error, Check Error, Data Late, and Error.

f=C Set the Busy, Done and all error indicating flags to 0 and stop all positioning and data transferring operations.

f=P Initiate either a Seek or a Recalibrate operation, depending on the contents of the command register.

SPECIFY DISC ADDRESS AND SECTOR COUNT

DOC<f> ac, DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	1	0	F		0	1	1	0	1	1

Bits 0-15 of the specified AC are loaded into the controller's combined Address Register/Sector Counter. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DRIVE		MAINT	SURFACE				SECTOR					-SECTOR COUNT			

Bits	Name	Function
0-1	Drive	Select the drive 0, 1, 2, or 3.
2	Format	If 1, place the drive in the format mode of operation.
3-6	Surface	Select the surface (head), 0-23_8, for the start of a Read or a Write operation.
7-11	Sector	Select the starting sector, 0-13_8, for the start of a Read or a Write operation.
12-15	-Sector Count	Specify the two's complement of the number of sectors to be read or written in one operation (maximum of 16).

READ STATUS

DIA<f> ac, DKP

0	1	1	AC		0	0	1	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Status Register are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

COMMAND FINISHED					SEEKING ON DRIVE										
READ/ WRITE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	0	1	2	3	DRIVE READY	SEEK ERROR	END ERROR	AD- DRESS ERROR	CHECK ERROR	DATA LATE	ERROR
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Meaning When 1
0	Read/Write Done	The subsystem has completed a Read or a Write operation.
1-4	Seek Done	Drive 0-3, respectively, has completed a Seek or Recalibrate operation. More than one of these bits can be set at any time.
5-8	Seeking On Drive	Drive 0-3, respectively, is currently performing a Seek or a Recalibrate operation.
9	Drive Ready	The selected drive is ready to carry out a command.
10	Seek Error	The selected drive did not carry out the Seek or the Recalibrate operation which was initiated.
11	End Error	The selected drive attempted to continue a Read or a Write operation beyond the last surface in the disc pack.
12	Address Error or Unsafe	The address read from the address field at the beginning of the track does not match the specified address; or a malfunction exists in the selected drive.
13	Check Error	The checkword read from the disc at the end of a sector does not match the checkword calculated by the controller.
14	Data Late	The data channel failed to respond in time to a data channel request.
15	Error	One or more of the bits 10-14 is set to 1.

SPECIFY COMMAND AND CYLINDER

DOA<f> ac, DKP

0	1	1	AC		0	1	0	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-15 of the specified AC are loaded into the controller's combined Command/Cylinder Select Register. Bit 5 is ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

CLEAR DONE FLAG						COMMAND		CYLINDER							
READ/ WRITE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	▨										
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function If Set To One
0	Clear Read/Write Done	Set the controller's Done flag to 0; set the following error indicating flags to 0: Address Error or Unsafe, End Error, Check Error and Seek Error.
1-4	Clear Seek Done	Set the Seek Done flags to 0 for the drives 0-3, respectively.
6-7	Command	Specify the command for the selected drive as follows: 00 Read with a Start command. 01 Write with a Start command. 10 Seek with a Pulse command to the cylinder specified in bits 8-15 of this accumulator. 11 Recalibrate with a Pulse command.
8-15	Cylinder	Specify the cylinder, 0-312_8, for a Seek, Read or a Write operation.

LOAD MEMORY ADDRESS COUNTER

DOB<f> ac, DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	0	0	F		0	1	1	0	1	1

Bits 0-15 of the specified AC are loaded into the controller's Memory Address Counter. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	MEMORY ADDRESS														

Bits	Name	Function
0	----	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

READ DISC ADDRESS

DIC<f> ac, DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	0	1	F		0	1	1	0	1	1

The contents of the disc Address Register and the Sector Counter are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DRIVE		FOR-MAT	SURFACE					SECTOR				-SECTOR COUNT			

Bits	Name	Contents
0-1	Drive	Number of the selected drive.
2	Format	The selected drive is in the format mode.
3-7	Surface	The surface number of the active head on the drive.
8-11	Sector	The number of the sector immediately following the last sector read or written.
12-15	-Sector Count	The two's complement of the number of sectors left to be read or written.

READ MEMORY ADDRESS COUNTER

DIB<f> ac, DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	1	F		0	1	1	0	1	1

The contents of the Memory Address Counter are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MEMORY ADDRESS															

Bits	Name	Contents
0-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

If bit 0 is 1, the Read or the Write operation exceeded the capacity of a 32K machine and the excess words were transferred either to (in a read) or from (in a write) low core.

PROGRAMMING

The preparation of a moving head disc pack for data channel transfers is divided into three distinct phases: I, selecting the drive, the surface and the sector; II, positioning the heads over the correct cylinder, and III, starting the Read or the Write operation. The results of issuing commands in each phase should be checked for errors before proceeding the next phase.

Phase I: Select the Drive, Surface, Sector and Number of Sectors

The initial selection of a disc drive is performed as follows: a SPECIFY DISC ADDRESS AND SECTOR COUNT instruction (DOC) is issued to the controller to select the drive, the surface of the disc pack, the first sector to be read or written, and the two's complement of the number of sectors to be transferred in the operation. The drives are numbered 0-3; the surfaces are numbered $0\text{-}23_8$; the sectors are numbered $0\text{-}13_8$; the maximum number of sectors which can be transferred in one operation is 16. Care should be taken to insure that the parameters specified in this initial selection do not exceed the capacity of the disc pack.

Once the drive unit is chosen, the status of that drive must be checked to determine if the drive is ready to proceed. The status is checked by issuing a READ STATUS instruction (DIA) and examining the Ready flag. If the Ready flag is set to 1, the program can proceed to Phase II. If it is set to 0, the program should not issue any commands to that drive unit until it is in the ready state.

Phase II: Position the Heads

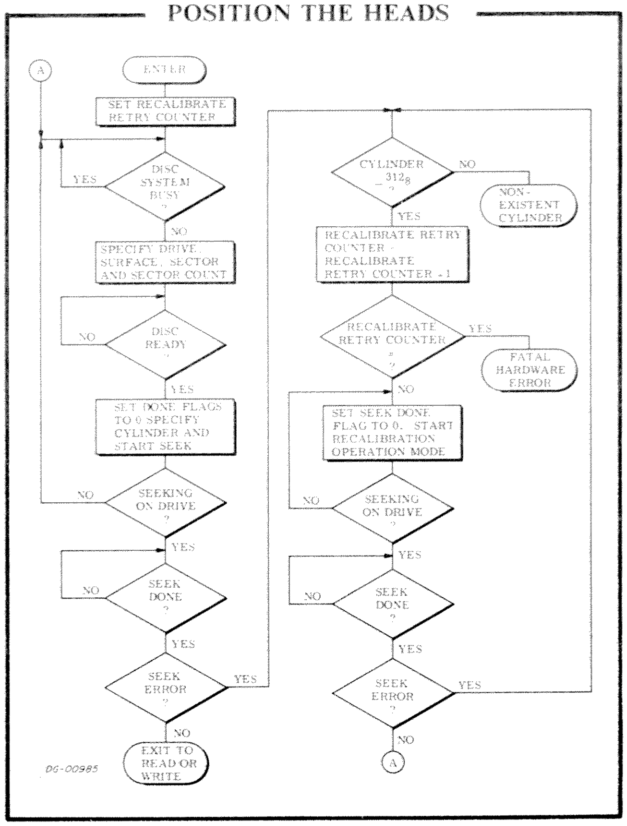
The heads are positioned over the desired cylinder as follows: a SPECIFY COMMAND AND CYLINDER instruction (DOA) is issued to the controller. This instruction should contain the number of the cylinder desired and the Seek command. The cylinders are numbered from 0-312$_8$. The instruction should also set both the DP Done flag and the Seek Done flag for the selected drive to 0. The Seek operation is initiated by a Pulse command. While the drive is seeking, the Unit Ready flag for that drive is set to 0 and the Seeking On Drive flag is set to 1. When the heads have finished moving to the specified cylinder, the Seeking On Drive flag is set to 0, the Unit Ready and the Seek Done flags for that drive unit are set to 1, thus initiating a program interrupt request.

The program should then check the Status Register to determine if a seek error has occurred as a result of a faulty Seek operation. If no errors have occurred, the program can proceed to Phase III.

The heads of the selected drive unit can be forced to cylinder 0 by the Recalibrate operation. A Recalibrate operation is performed as follows: a SPECIFY COMMAND AND CYLINDER instruction is issued to the controller. This instruction should contain the Recalibrate command and should also set both the DP Done and the Seek Done flags for the selected drive unit to 0. The operation is initiated by a Pulse command. While the drive is being recalibrated, Unit Ready flag for the selected drive is set to 0 and the Seeking On Drive flag is set to 1. Once the Recalibrate operation is completed, the Seeking On Drive flag is set to 0, the Unit Ready flag is set to 1, the Seek Done flag for the selected drive is set to 1 and a program interrupt request is initiated.

When the program places a drive in the seek mode of operation, the controller is free to accept commands to the other drives under its direction. Therefore, once one or more drives are performing Seek operations, one of the other drives can perform a Read or a Write operation. If the program is simultaneously managing several drives with one controller, the error indicating flags in the Status Register apply only to the most recently selected drive unit, i. e. , the most specified in the last SPECIFY DISC ADDRESS AND SECTOR COUNT instruction issued.

POSITION THE HEADS


A
ENTER
SET RECALIBRATE RETRY COUNTER
DISC SYSTEM BUSY ?
YES
NO
SPECIFY DRIVE, SURFACE, SECTOR AND SECTOR COUNT
DISC READY ?
NO
YES
SET DONE FLAGS TO 0 SPECIFY CYLINDER AND START SEEK
SEEKING ON DRIVE ?
NO
YES
SEEK DONE ?
NO
YES
SEEK ERROR ?
YES
NO
EXIT TO READ OR WRITE
DG-00985
CYLINDER ≤ 312$_8$?
NO
NON-EXISTENT CYLINDER
YES
RECALIBRATE RETRY COUNTER = RECALIBRATE RETRY COUNTER -1
RECALIBRATE RETRY COUNTER = ?
YES
FATAL HARDWARE ERROR
NO
SET SEEK DONE FLAG TO 0. START RECALIBRATION OPERATION MODE
SEEKING ON DRIVE ?
NO
YES
SEEK DONE ?
NO
YES
SEEK ERROR ?
YES
NO
A

Phase III: Read or Write

A Read operation transfers blocks of data stored in the disc subsystem to the computer, via the data channel. A block of data contains 256 16-bit words. Up to 16 blocks can be transferred in one Read operation. A Write operation transfers blocks of data from the computer's memory, via the data channel, and stores the data in the disc subsystem. Again, up to 16 blocks of data may be transferred in one operation.

Read or Write operations are performed in a series of steps which are virtually identical. The parameters of the data transfer must be specified and finally the operation is initiated.

Read

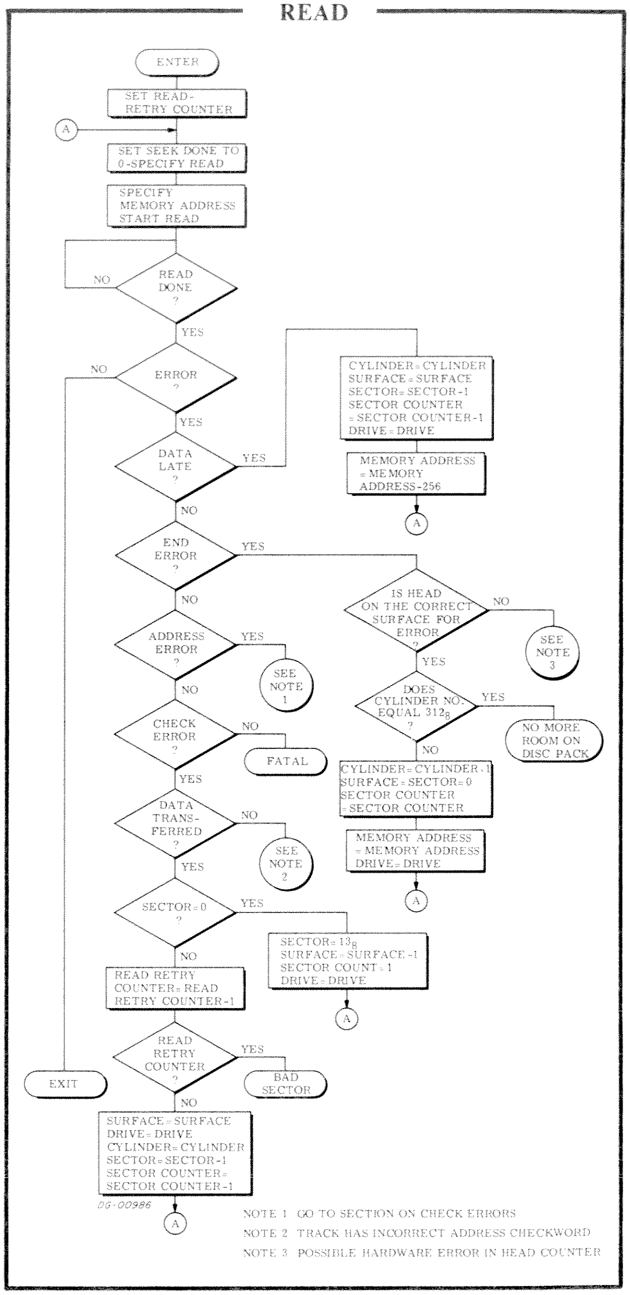
A Read operation is performed as follows: the storage location in memory for the first word to be read from the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be read and the starting sector were specified in Phase I.

The Read command is then loaded into the Command Register with a SPECIFY COMMAND AND CYLINDER instruction (DOA). It is not necessary to load the cylinder number again, but it is good practice since other disc drives may require this information. The Read operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

Once the Read operation is initiated, the drive waits until the address field for the track passes under the head and performs an address check. If the address read is correct, the controller waits for the starting sector and when it is encountered, the drive starts reading the sequential bits of the first word of the sector. When a word is fully assembled, the controller transfers the word to the computer's memory via the data channel. Each time a word is transferred to memory, the Memory Address Counter is automatically incremented.

When the 256 words from the sector have been read, and the checkword at the end of the sector verified, the sector counter is automatically incremented by one.

READ


ENTER
SET READ-RETRY COUNTER
A
SET SEEK DONE TO 0-SPECIFY READ
SPECIFY MEMORY ADDRESS START READ
NO
READ DONE ?
YES
NO
ERROR ?
YES
DATA LATE ?
YES
CYLINDER = CYLINDER SURFACE = SURFACE SECTOR = SECTOR-1 SECTOR COUNTER = SECTOR COUNTER-1 DRIVE = DRIVE
MEMORY ADDRESS = MEMORY ADDRESS-256
A
NO
END ERROR ?
YES
IS HEAD ON THE CORRECT SURFACE FOR ERROR ?
NO
SEE NOTE 3
YES
DOES CYLINDER NO. EQUAL 312₈ ?
YES
NO MORE ROOM ON DISC PACK
NO
CYLINDER = CYLINDER+1 SURFACE = SECTOR = 0 SECTOR COUNTER = SECTOR COUNTER
MEMORY ADDRESS = MEMORY ADDRESS DRIVE = DRIVE
A
NO
ADDRESS ERROR ?
YES
SEE NOTE 1
NO
CHECK ERROR ?
NO
FATAL
YES
DATA TRANS-FERRED ?
NO
SEE NOTE 2
YES
SECTOR = 0 ?
YES
SECTOR = 13₈ SURFACE = SURFACE -1 SECTOR COUNT + 1 DRIVE = DRIVE
A
NO
READ RETRY COUNTER = READ RETRY COUNTER-1
READ RETRY COUNTER ?
YES
BAD SECTOR
EXIT
NO
SURFACE = SURFACE DRIVE = DRIVE CYLINDER = CYLINDER SECTOR = SECTOR-1 SECTOR COUNTER = SECTOR COUNTER-1
A

DG-00986

NOTE 1 GO TO SECTION ON CHECK ERRORS

NOTE 2 TRACK HAS INCORRECT ADDRESS CHECKWORD

NOTE 3 POSSIBLE HARDWARE ERROR IN HEAD COUNTER

If the sector counter does not overflow, the next sector is read and this process continues until either the sector counter overflows or the last sector on the surface is read. In the case where the last sector on surface 0-22_8 is read, and the sector counter has not overflowed, the drive will automatically continue the operation by reading the first sector on the next surface in the same cylinder.

The Read operation then continues until the sector counter indicates, by overflowing, that the specified number of sectors have been read. Upon completion of the Read operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

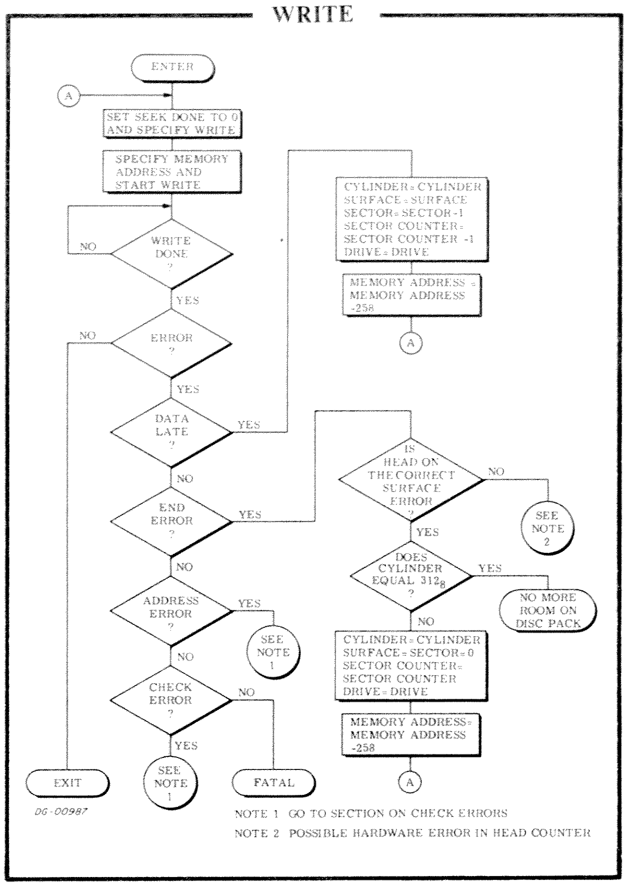
Write

A Write operation is performed as follows: the storage location in memory of the first word to be written on the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be written and the number of the starting sector were specified in Phase I. The Write command is then loaded into the Command Register with a SPECIFY COMMAND AND CYLINDER instruction (DOA). It is not necessary to load the cylinder number again, but it is good practice since other disc drives may require this information. The Write operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

Once the Write operation is initiated, the controller reads two words from the computer's memory via the data channel and then waits for the address field for the track to pass under the head. Each time the controller reads a word from the computer's memory, the Memory Address Counter is automatically incremented. Once the address field is encountered, the address check is performed automatically. If the address read from the disc is correct, the controller waits for the starting sector and when it is encountered, the bits of each word are sequentially written. When the 256 words in the sector have been written, the controller writes the checkword it calculated from the data it wrote on the disc. The sector counter is then automatically incremented by one.

If the sector counter does not overflow, the next sector is written. This process continues until either the sector counter overflows or the last sector on the surface is written. In the case when the last sector on surface 0-22_8 is written, and the sector counter has not overflowed, the drive will automatically continue the operation by writing the first sector on the next surface in the same cylinder.

The Write operation then continues until the sector counter indicates by overflowing, that the specified number of sectors have been written. Upon completion of the Write operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt is initiated.


WRITE

ENTER
A
SET SEEK DONE TO 0 AND SPECIFY WRITE
SPECIFY MEMORY ADDRESS AND START WRITE
WRITE DONE ?
NO
YES
ERROR ?
NO
YES
DATA LATE ?
YES
NO
END ERROR ?
YES
NO
ADDRESS ERROR ?
YES
SEE NOTE 1
NO
CHECK ERROR ?
NO
YES
SEE NOTE 1
EXIT
FATAL
CYLINDER = CYLINDER
SURFACE = SURFACE
SECTOR = SECTOR -1
SECTOR COUNTER = SECTOR COUNTER -1
DRIVE = DRIVE
MEMORY ADDRESS = MEMORY ADDRESS -258
A
IS HEAD ON THE CORRECT SURFACE ERROR ?
NO
SEE NOTE 2
YES
DOES CYLINDER EQUAL 312₈ ?
YES
NO MORE ROOM ON DISC PACK
NO
CYLINDER = CYLINDER
SURFACE = SECTOR = 0
SECTOR COUNTER = SECTOR COUNTER
DRIVE = DRIVE
MEMORY ADDRESS = MEMORY ADDRESS -258
A
DG-00987

NOTE 1 GO TO SECTION ON CHECK ERRORS

NOTE 2 POSSIBLE HARDWARE ERROR IN HEAD COUNTER

Format

A new disc pack is unformatted so a Disc Pack Formatter program is available to initialize the disc pack with the proper prerequisites for reading from or writing on the disc pack.

This program is the 4057 Disc Pack Formatter Program #096-000038 (tape #095-000071).

A disc pack may also be reformatted when repeated recalibrations of the heads do not eliminate an Address Error occurring during a Read or a Write operation. All data is lost when a disc pack is reformatted.

TIMING

The disc pack rotates at a speed of 2400rpm; a complete revolution requires 25 milliseconds. Since the controller must wait until the index point on the disc passes the head before the search for the desired sector can be started, the latency time before starting the sector search can be as much as 25 milliseconds (average of 12. 5 milliseconds. Once the search for the correct sector begins, the time taken to reach that sector depends on which sector is specified: sector 0 is reached almost immediately while sector 13_8 requires 23. 2 milliseconds before reading or writing can commence.

The total sector access time is therefore the time required to reach the address field plus the time required to reach the desired sector. The maximum is 48. 2 milliseconds (25 to reach the address field plus the 23. 2 milliseconds to reach sector 13_8).

The time required to position the heads (the seek time) is dependent on the number of cylinders the heads must move past in a Seek operation. A maximum of 10 milliseconds is required to move the heads from one cylinder to the adjacent cylinder. The time required to move from cylinder 0 to cylinder 312_8, or vice versa, is 60 milliseconds, maximum. A Recalibrate operation requires a maximum of 300 milliseconds.

If a Seek or a Recalibrate operation is not completed within 1 second, the Seek Error and Error flags are set to 1. The Done flag is set to 1 and a program interrupt request is initiated.

A sector passes under the head in 2. 08 milliseconds while the data block in the sector passes under the head in 1. 64 milliseconds. Since there are 256 data words in a sector, a data channel request occurs every 6. 4 microseconds. This corresponds to a data transfer rate of 156, 000 words/second. Since the controller is double buffered, the maximum allowable data channel latency is 12. 8 microseconds. If the data channel does not respond within this time, both the Data Late and the Error flags are set to 1. Once this error occurs, the processing of the current sector is completed and the command is terminated, even if the operation was scheduled to transfer additional sectors. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

For the greatest efficiency when processing an entire cylinder, the program should be coded to process only one track at a time since the head must wait for the index point on the surface to pass the head before each command can begin. However, after completing the processing of a track, enough time is available (. 2msec) to prepare the disc pack to read or write the next track without waiting for a full rotation of the disc to correctly position sector zero. This procedure allows the head to start processing immediately after a command is issued without having to wait for the drive to search for the correct sector before beginning the operation.

ERROR CONDITIONS

Durinş Initial Selection

If the program specifies a non-existent sector ($<13_8$) no indication of this error is given. If the subsystem then attempts a Read or a Write operation, the drive unit will search forever for that non-existent sector. The subsystem can be released from this search by having the program issue an I/O RESET instruction (IORST).

During Head Positioning

If the program issues a SPECIFY COMMAND AND CYLINDER instruction which specifies a non-existent cylinder ($<312_8$) and then places the drive unit in the seek mode, the Seek operation is terminated. Both the Seek Error and the Seek Done flags for that unit are set to 1. When the Seek Done flag is set to 1, a program interrupt request is initiated.

If any Seek operation to a valid cylinder number results in a Seek Error, the drive unit should be recalibrated.

During Reading

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Read operation is initiated.

An error can occur when the subsystem is reading a series of sectors in one operation if the Read operation exceeds the number of sectors available. When the last sector on surface 23_8 is read and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Each track on the disc pack is preceded by an address field. If the address read before any track does not match the contents of the disc address registers at that time, both the Error and the Address Error flags are set to 1. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The reading of the current sector continues, but once that sector has been read, the Read operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word from the last sector read was not correctly transferred to memory.

During Writing

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Write operation is initiated.

An error can occur when the subsystem is writing a series of sectors in one operation if the Write operation exceeds the number of sectors available. When the last sector on surface 23_8 is written and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to sector 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Each track on the disc pack is preceded by an address field. If the address read before any track does not match the contents of the disc address registers at that time, both the Error and the Address Error flags are set to 1. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The writing of the current sector continues, but once that sector has been written, the Write operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word in the sector was not written properly.

Check Error

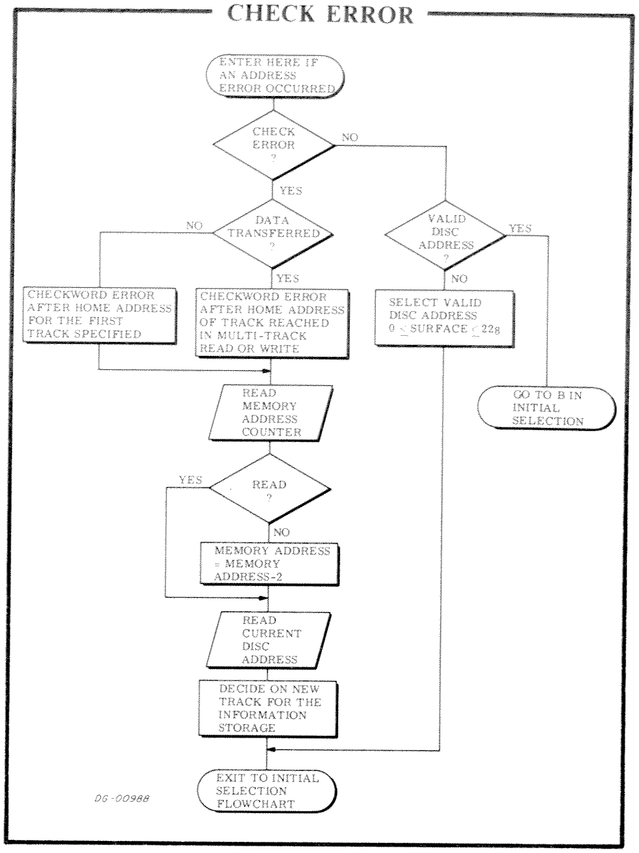
A Check Error flag is set to 1 during a Read operation if the 16-bit checkword read after the data in the sector being processed does not match the checkword calculated by the controller during the Read operation.

The Check Error flag can also be set to 1 if a checkword error is detected when verifying the home address of a track in either a Read or a Write operation. Usually the Address Error flag is also set to 1 since an error has been made in reading the track address information. However, this Checkword Error can also occur without an accompanying Address Error when the checkword read at the end of the track home address is incorrect. In both of these cases, access to all the sectors on that track is denied.

Since the Check Error flag can be set to 1 without an Address Error flag under two circumstances in a Read operation, the programmer should verify which condition caused the flag to be set to 1. If the Check Error flag is posted and the Memory Address Counter has not changed, the checkword following the track home address is incorrect. In addition, if the Read operation does not extend to the next surface and data has been transferred, then the error is in the last sector processed.

However, when a Read operation extends beyond one surface, and a Check Error occurs after the last sector on the first surface is read and before the first sector on the second surface is read, the error could be in either the last sector on the first surface or in the checkword for the home address of the track in the second surface. In each of these two cases, the Memory Address Counter and the Disc Address Register will contain identical information.

CHECK ERROR


ENTER HERE IF AN ADDRESS ERROR OCCURRED
CHECK ERROR ?
NO
YES
DATA TRANSFERRED ?
NO
YES
CHECKWORD ERROR AFTER HOME ADDRESS FOR THE FIRST TRACK SPECIFIED
CHECKWORD ERROR AFTER HOME ADDRESS OF TRACK REACHED IN MULTI-TRACK READ OR WRITE
VALID DISC ADDRESS ?
YES
NO
SELECT VALID DISC ADDRESS 0 ≤ SURFACE ≤ 22₈
GO TO B IN INITIAL SELECTION
READ MEMORY ADDRESS COUNTER
READ ?
YES
NO
MEMORY ADDRESS = MEMORY ADDRESS-2
READ CURRENT DISC ADDRESS
DECIDE ON NEW TRACK FOR THE INFORMATION STORAGE
EXIT TO INITIAL SELECTION FLOWCHART
DG-00988

These two cases can be differentiated by performing a Read operation on the last sector of the first surface. If this Read is successful, the error is in the checkword of the home address of the track on the second surface. If this is the case, then that track on the second surface is inaccessible.

This page intentionally left blank

THE 4231A DISC PACK SUBSYSTEM

SUMMARY

MNEMONIC (FIRST CONTROLLER)	DKP
DEVICE CODE (FIRST CONTROLLER)	33_8
MNEMONIC (SECOND CONTROLLER)	DKP1
DEVICE CODE (SECOND CONTROLLER)	73_8
PRIORITY MASK BIT	7
SURFACES/UNIT	19
TRACKS/SURFACE (CYLINDERS)	411
SECTORS/TRACK	23
WORDS/SECTOR	256
TOTAL STORAGE CAPACITY (WORDS)	45,979,392
MAXIMUM TRANSFER RATE (WORDS/SEC)	403,000
ALLOWABLE DATA CHANNEL LATENCY (μSEC)	19.8
SEEK TIME MAX/MIN (mSEC)	55/10
SECTOR ACCESS TIME MAX/MIN (mSEC)	16.7/0

ACCUMULATOR FORMATS

SPECIFY DISC ADDRESS AND SECTOR COUNT .... (DOC)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DRIVE		SURFACE					SECTOR					-SECTOR COUNT			

READ DISC ADDRESS .... (DIC)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DRIVE		SURFACE					SECTOR					-SECTOR COUNT			

SPECIFY COMMAND AND CYLINDER .. (DOA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CLEAR DONE FLAG															
DP DONE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	COMMAND		CYLINDER								

COMMANDS

00	Read	10	Seek
01	Write	11	Recalibrate

READ STATUS .... (DIA)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
COMMAND DONE															
DP DONE	SEEK 0	SEEK 1	SEEK 2	SEEK 3	DUAL PROC	SECTOR ERROR	HEAD ERROR	ADDRESS ERROR	DISC READY	SEEK ERROR	END ERROR	UNSAFE	CHECK ERROR	DATA LATE	ERROR

LOAD MEMORY ADDRESS COUNTER ... (DOB)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FORMAT	MEMORY ADDRESS														

READ MEMORY ADDRESS COUNTER .... (DIB)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BAD SECTOR	MEMORY ADDRESS														

S, C AND P FUNCTIONS

S Set Busy to one, Done to zero and start a Read or a Write operation.

C Set Busy to zero, Done to zero and stop all operations.

P Start a Seek or a Recalibrate operation.

INTRODUCTION

The 4231A disc pack subsystem utilizes a removable disc pack containing 19 (0-22_8) program accessible surfaces. There are 411 (0-632_8) cylinders on the disc pack. Each of the 19 tracks in a cylinder contains 23 (0-26_8) performatted sectors, each of which stores 256 (400_8) 16-bit words and contains a checkword. The data storage capacity is 5888 words/track, 111,872 words/cylinder or 45,979,392 words/pack. Words are transferred to and from the subsystem via the data channel at a rate of 403,000 words per second. Up to 16 sectors containing 4096 (10000_8) words can be transferred in one operation.

The controller for the subsystem, when coupled to the adapter, can direct the activities of up to four drive units. Any number of these units can be performing Seek operations simultaneously, but only one drive can be reading or writing at any one time.

INSTRUCTIONS

The disc drive controller contains four program accessible registers: a 15-bit Memory Address Counter, a 16-bit Status Register, a 16-bit combined Command/Cylinder Select Register and a combined disc Address/Sector Counter Register. The Memory Address Counter is self-incrementing and contains the memory location of the next 16-bit word to be either read from or written on the disc. The Status Register contains the information flags for the controller and the selected drive as well as 4 flags which indicate when any drive completes a Seek or a Recalibrate operation. Any of these 4 Seek Done flags as well as Disc Pack Done flag are able to initiate a program interrupt request when they are set to 1. The combined Command/Cylinder Select Register contains the command last issued to the subsystem and the number of the desired cylinder on the disc surface. The combined Disc Address/Sector Counter contains the surface and sector location of the active head and the two's complement of the number of sectors to be either read from or written on the disc. The Sector Counter is self-incrementing after each sector is read or written.

Six instructions are used to program data channel transfers to and from the disc pack. Three of these instructions are used to supply all of the necessary data to the controller for any disc operation. The remaining three instructions allow the program to determine, in detail, the current state of the disc pack subsystem.

The disc controller's Busy and Done flags are controlled using all three of the device flag commands as follows:

f=S Set the Busy flag to 1, the Done and the following error indicating flags to 0, and initiate a Read or a Write operation, depending on the contents of the command register. The error indicating flags are: Sector Error, Head Error, Address Error, Seek Error, End Error, Check Error, Data Late, Bad Sector, and Error.

NOTE The Unsafe flag can be set to 0 only with an I/O RESET instruction (IORST).

f=C Set the Busy, Done and all error indicating flags to 0 and stop all positioning and data transferring operations.

f=P Initiate either a Seek or a Recalibrate operation, depending on the contents of the command register.

SPECIFY DISC ADDRESS AND SECTOR COUNT

DOC<f> ac,DKP

0	1	1	AC		1	1	0	F		0	1	1	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

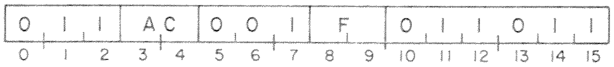
Bits 0-15 of the specified AC are loaded into the controller's combined Address Register/Sector Counter. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

DRIVE		SURFACE					SECTOR					-SECTOR COUNT			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

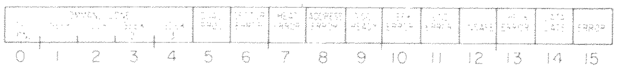
Bits	Name	Function
0-1	Drive	Select the drive 0, 1, 2, or 3.
2-6	Surface	Select the surface (head) 0-22_8, for the start of a Read or a Write operation.
7-11	Sector	Select the starting sector, 0-26_8 for the start of a Read or a Write operation.
12-15	-Sector Count	Specify the two's complement of the number of sectors to be read or written in one operation (maximum of 16).

READ STATUS

DIA<f> ac, DKP



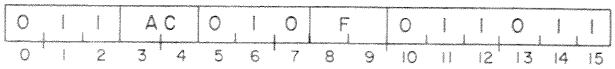
The contents of the Status Register are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:



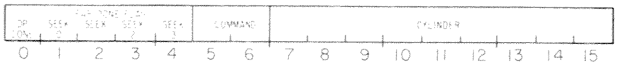
Bits	Name	Meaning When 1
0	DP Done	The subsystem has completed a Read or a Write operation.
1-4	Seek Done	Drive 0-3,. respectively, has completed a Seek or Recalibrate operation. More than one of these bits can be set at any time.
5	Dual Processor	The disc subsystem is in a dual processor configuration.
6	Sector Error	Must appear in combination with bit 8, Address Error. The sector address portion of the address field is incorrect.
7	Head Error	Must appear in combination with bit 8, Address Error. The head address portion of the address field is incorrect.
8	Address Error	The address read from the address field at the beginning of a sector does not match the last address specified to the disc controller. If this bit is 1 and bits 6 and 7 are both 0, the cylinder address portion of the address field is incorrect.
9	Disc Ready	The selected drive is not performing any head movements and is ready to carry out a command.
10	Seek Error	The selected drive did not successfully carry out the Seek or Recalibrate operation which was initiated.
11	End Error	The selected drive attempted to continue a Read or a Write operation beyond the last surface in the disc pack.
12	Unsafe	The selected drive is in an unsafe condition. An IORST must be issued.
13	Check Error	The checkword read from the disc at the end of a sector does not match the checkword calculated by the controller.
14	Data Late	The data channel failed to respond in time to a data channel request.
15	Error	One or more of the bits 8, 10, 11, 12, 13, 14 in this Status Register, or the Bad Sector flag is set to 1.

SPECIFY COMMAND AND CYLINDER

DOA<f> ac, DKP



Bits 0-15 of the specified AC are loaded into the controller's combined Command/Cylinder Select Register. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:



Bits	Name	Function
0	Clear DP Done	Set the controller's Done flag to 0; set the following error indicating flags to 0: Address Error, End Error, Check Error, and Bad Sector.
1-4	Clear Seek Done	Set the Seek Done flags to 0 for the drives 0-3, respectively.
5-6	Command	Specify the command for the selected drive as follows: 00 Read with a Start command. 01 Write with a Start command. 10 Seek with a Pulse command to the cylinder specified in bits 7-15 of this accumulator. 11 Recalibrate with a Pulse command.
7-15	Cylinder	Specify the cylinder, $0\text{-}632_8$, for a Seek, Read or Write operation.

LOAD MEMORY ADDRESS COUNTER

DOB<f> ac,DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	0	0	F		0	1	1	0	1	1

Bits 0-15 of the specified AC are loaded into the controller's Memory Address Counter. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FORMAT	MEMORY ADDRESS														

Bits	Name	Function
0	Format	If 1, place the drive in the format mode of operation.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

READ DISC ADDRESS

DIC<f> ac,DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		1	0	1	F		0	1	1	0	1	1

The contents of the disc Address Register and the Sector Counter are placed in bits 0-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DRIVE		SURFACE					SECTOR					-SECTOR COUNT			

Bits	Name	Contents
0-1	Drive	Number of the selected drive.
2-6	Surface	The surface number of the active head on the drive.
7-11	Sector	The number of the sector immediately following the last sector read or written.
12-15	-Sector Count	The two's complement of the number of sectors left to be read or written.

READ MEMORY ADDRESS COUNTER

DIB<f> ac,DKP

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	AC		0	1	1	F		0	1	1	0	1	1

The contents of the Memory Address Counter are placed in bits 1-15 of the specified AC. A Bad Sector indicator flag is placed in bit 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BAD SECTOR	MEMORY ADDRESS														

Bits	Name	Contents
0	Bad Sector	If 1, the sector was found to be unusable during formatting.
1-15	Memory Address	Location of the next word in memory to be used for a data channel transfer.

The preparation of a moving head disc pack for data channel transfers is divided into three distinct phases: I, selecting the drive, the surface and the sector; II, positioning the heads over the correct cylinder; and III, starting the Read or the Write operation. The results of issuing commands in each phase should be checked for errors before proceeding to the next phase.

Phase I: Select the Drive, Surface, Sector and Number of Sectors

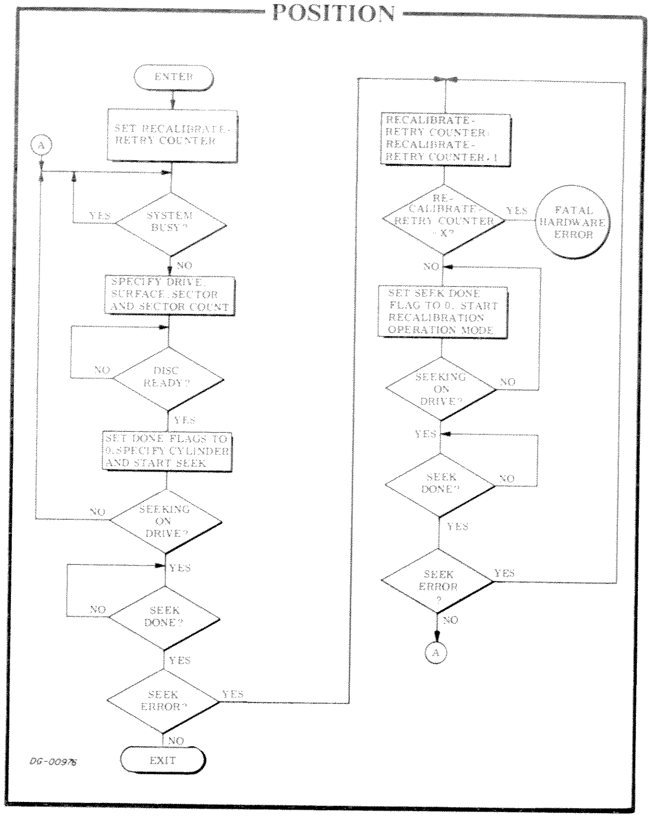
The initial selection of a disc drive is performed as follows: a SPECIFY DISC ADDRESS AND SECTOR COUNT instruction (DOC) is issued to the controller to select the drive, the surface of the disc pack, the first sector to be read or written, and the two's complement of the number of sectors to be transferred in the operation. The drives are numbered 0-3; the surfaces are numbered 0-22_8; the sectors are numbered 0-26_8; the maximum number of sectors which can be transferred in one operation is 16. Care should be taken to insure that the parameters specified in this initial selection do not exceed the capacity of the disc pack.

Once the drive unit is chosen, the status of that drive must be checked to determine if the drive is ready to proceed. The status is checked by issuing a READ STATUS instruction (DIA) and examining the Ready flag. If the Ready flag is 1, the program can proceed to Phase II, If it is 0, the program should not issue any commands to that drive unit until it is in the ready state.

Phase II: Position the Heads

The heads are positioned over the desired cylinder as follows: a SPECIFY COMMAND AND CYLINDER instruction (DOA) is issued to the controller. This instruction should contain the number of the cylinder desired and the Seek command. The cylinders are numbered from 0-632_8. The instruction should also set both the DP Done flag and the Seek Done flag for the selected drive to 0. The Seek operation is initiated by a Pulse command. While the drive is seeking, the Unit Ready flag for that drive is set to 0. When the heads have finished moving to the specified cylinder, the Unit Ready and the Seek Done flags for that drive unit are set to 1, thus initiating a program interrupt request.

The program should then check the Status Register to determine if a seek error has occurred as a result of a faulty Seek operation. If no errors have occurred, the program can proceed to Phase III.


POSITION
ENTER
SET RECALIBRATE-RETRY COUNTER
A
SYSTEM BUSY?
YES
NO
SPECIFY DRIVE, SURFACE, SECTOR AND SECTOR COUNT
DISC READY?
NO
YES
SET DONE FLAGS TO 0, SPECIFY CYLINDER AND START SEEK
SEEKING ON DRIVE?
NO
YES
SEEK DONE?
NO
YES
SEEK ERROR?
YES
NO
EXIT
DG-00976
RECALIBRATE-RETRY COUNTER: RECALIBRATE-RETRY COUNTER, 1
RE-CALIBRATE-RETRY COUNTER =X?
YES
FATAL HARDWARE ERROR
NO
SET SEEK DONE FLAG TO 0, START RECALIBRATION OPERATION MODE
SEEKING ON DRIVE?
NO
YES
SEEK DONE?
NO
YES
SEEK ERROR ?
YES
NO
A

The heads of the selected drive unit can be forced to cylinder 0 by the Recalibrate operation. A Recalibrate operation is performed as follows: a SPECIFY COMMAND AND CYLINDER instruction is issued to the controller. This instruction should contain the Recalibrate command and should also set both the DP Done and the Seek Done flags for the selected drive unit to 0. The operation is initiated by a Pulse command. While the drive is being recalibrated, the Unit Ready flag for the selected drive is set to 0. Once the Recalibrate operation is completed, the Unit Ready flag is set to 1, the Seek Done flag for the selected drive is set to 1 and a program interrupt request is initiated.

When the program places a drive in the seek mode of operation, the controller is free to accept commands to the other drives under its direction. Therefore, once one or more drives are performing Seek operations, one of the other drives can perform a Read or a Write operation. If the program is simultaneously managing several drives with one controller, the error indicating flags in the Status Register apply only to the most recently selected drive unit, i. e., the unit specified in the last SPECIFY DISC ADDRESS AND SECTOR COUNT instruction issued.

Phase III: Read or Write

A Read operation transfers blocks of data stored in the disc subsystem to the computer, via the data channel. A block of data contains 256 16-bit words. Up to 16 blocks can be transferred in one Read operation. A Write operation transfers blocks of data from the computer's memory, via the data channel, and stores the data in the disc subsystem. Again, up to 16 blocks of data may be transferred in one operation. Read or Write operations are performed in a series of steps which are virtually identical. The parameters of the data transfer must be specified and finally the operation is initiated.

Read

A Read operation is performed as follows: the storage location in memory for the first word to be read from the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be read and the starting sector were specified in Phase I.

The Read command is then loaded into the Command Register with a SPECIFY COMMAND AND CYLINDER instruction (DOA). The cylinder number must be specified in this instruction since it is used in the address check which is automatically made before each 256 word sector is read. The Read operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

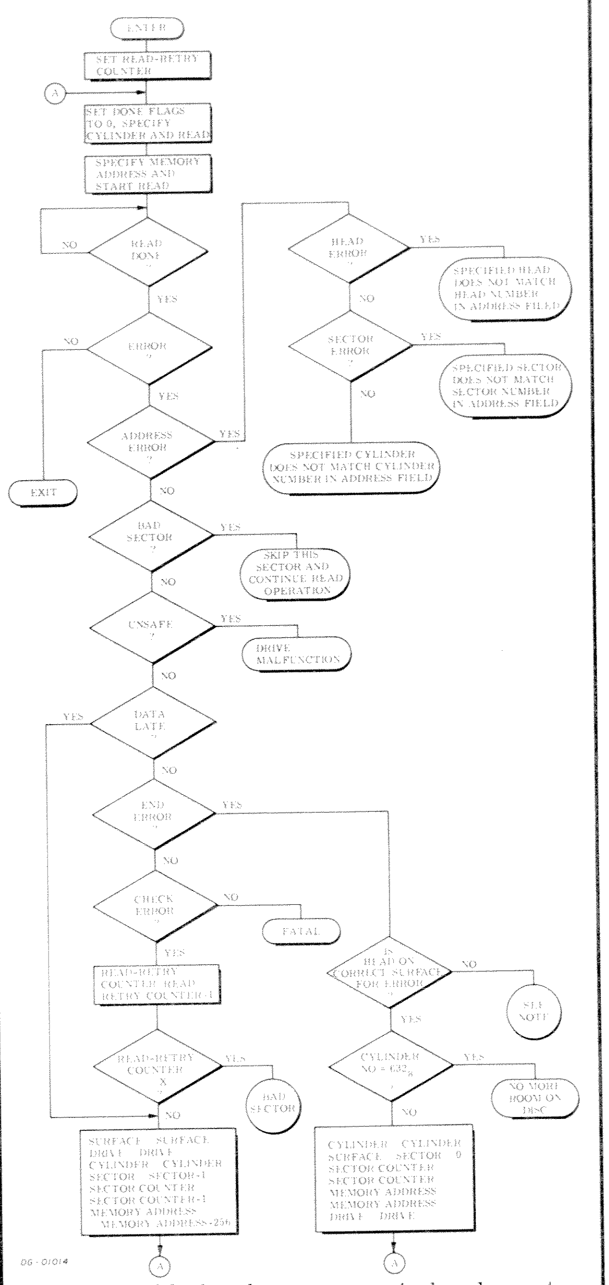
Once the Read operation is initiated, the drive waits until the desired sector passes under the head and performs an address check. If the address read is correct, the drive then starts reading the sequential bits of the first word of the sector. When a word is fully assembled, the controller transfers the word to the computer's memory via the data channel. Each time a word is transferred to memory, the Memory Address Counter is automatically incremented.

When the 256 words from the sector have been read, and the checkword at the end of the sector verified, the sector counter is automatically incremented by one.

If the sector counter does not overflow, the next sector is read and this process continues until either the sector counter overflows or the last sector on the surface is read. In the case where the last sector on surface $0\text{-}21_8$ is read, and the sector counter has not overflowed, the drive will automatically continue the operation by reading the first sector on the next surface in the same cylinder.

The Read operation then continues until the sector counter indicates, by overflowing, that the specified number of sectors have been read. Upon completion of the Read operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

READ



Note: Possible hardware error in head counter.

Write

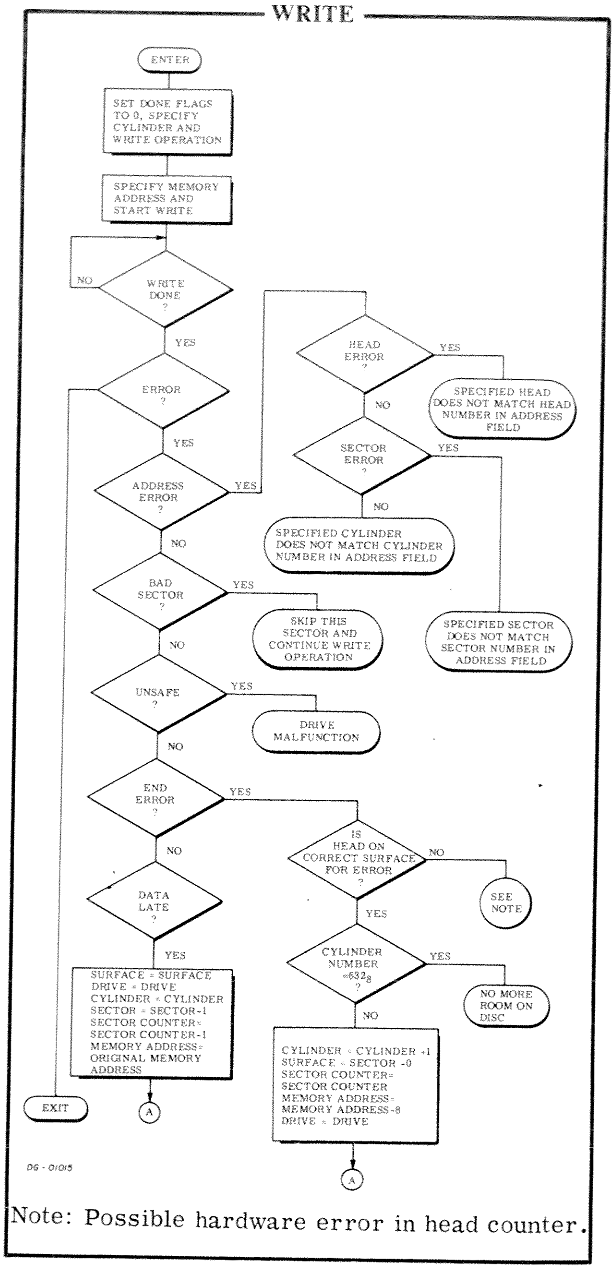
A Write operation is performed as follows: the storage location in memory of the first word to be written on the disc is specified with a LOAD MEMORY ADDRESS COUNTER instruction (DOB). The number of sectors to be written and the number of the starting sector were specified in Phase I.

The Write command is then loaded into the Command Register with a SPECIFY COMMAND AND CYLINDER instruction (DOA). The cylinder number must be specified in this instruction since it is used in the address check which is automatically made before each 256 word sector is written. The Write operation is initiated with a Start command. The Busy flag is set to 1 and the Done flag is set to 0.

Once the Write operation is initiated, the controller reads eight words from the computer's memory via the data channel and then waits for the desired sector to pass under the head. Each time the controller reads a word from the computer's memory, the Memory Address Counter is automatically incremented. Once the desired sector is encountered, the address check is performed automatically. If the address read from the disc is correct, the bits of each word are sequentially written. When the 256 words in the sector have been written, the controller writes the checkword it calculated from the data it wrote on the disc. The sector counter is automatically incremented by one.

If the sector counter does not overflow, the next sector is written. This process continues until either the sector counter overflows of the last sector on the surface is written. In the case when the last sector on surface 0-21$_8$ is written, and the sector counter has not overflowed, the drive will automatically continue the operation by writing the first sector on the next surface in the same cylinder.

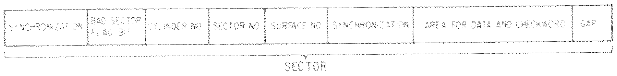
The Write operation then continues until the sector counter indicates, by overflowing, that the specified number of sectors have been written. Upon completion of the Write operation, the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt is initiated.



Note: Possible hardware error in head counter.

FORMAT

The disc pack used in the 4231A subsystem must be formatted so that the address check, made before every sector is read or written, can be carried out. If the disc pack is not formatted, an Address Error occurs and terminates every Read or Write operation. When the disc pack is formatted, the following information is written before every sector on the disc pack.



If a disc pack repeatedly indicates address errors, the pack may be reformatted. All data on a disc pack is lost when it is reformatted.

The program is the 4231 Disc Pack Formatter Program #096-000241 (tape #095-000221).

TIMING

The disc pack rotates at a speed of 3600rpm; a complete revolution requires 16.67 milliseconds. A register, containing the identification number of the sector currently passing under the head, is used to reduce the sector accessing time. This feature allows the subsystem to carry out a Read or a Write operation the first time the desired sector passes under the head.

Since a sector is preceded by a 68 microsecond gap, used for address checking, the minimum sector access time is 68 microseconds. The maximum is 16.7 milliseconds so the average is 8.4 milliseconds.

The time required to position the heads (the seek time) is dependent on the number of cylinders the heads must move past in a Seek operation. A maximum of 10 milliseconds is required to move the heads from one cylinder to the adjacent cylinder. The time required to move from cylinder 0 to cylinder 632_8, or vice versa, is 55 milliseconds, maximum.

A Recalibrate operation requires a maximum of 300 milliseconds. If either a Seek operation or a Recalibrate operation is not completed in 500 milliseconds, the Seek Error flag is set to 1.

A sector passes under the head in 730 microseconds while the data block in the sector passes under the head in 635 microseconds. Since there are 256 data words in a sector, a data channel request occurs every 2.48 microseconds. This corresponds to a data transfer rate of 403,000 words/second. Since the controller has an eight word buffer, the maximum allowable data channel latency is 19.8 microseconds. If the data channel does not respond within this time, both the Data Late and the Error flags are set to 1. Once this error occurs, the processing of the current sector is completed and the command is terminated, even if the operation was scheduled to transfer additional sectors. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

ERROR CONDITIONS

During Initial Selection

If the Unsafe flag is 1, the Unit Ready flag is 0. An I/O RESET instruction (IORST) is the only way the program can clear this flag.

If the program specifies a non-existent sector ($>26_8$) no indication of this error is given. If the subsystem then attempts a Read or a Write operation, the drive unit will search forever for that non-existent sector. The subsystem can be released from this search by having the program issue an I/O RESET instruction.

During Head Positioning

If the program issues a SPECIFY COMMAND AND CYLINDER instruction which specifies a non-existent cylinder ($>632_8$) and then places the drive unit in the seek mode, the Seek operation is terminated. Both the Seek Error and the Seek Done flags for that unit are set to 1. When the Seek Done flag is set to 1, a program interrupt request is initiated.

If any Seek operation to a valid cylinder number results in a Seek Error, the drive unit should be recalibrated.

During Reading

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Read operation is initiated.

An error can occur when the subsystem is reading a series of sectors in one operation if the Read operation exceeds the number of sectors available. When the last sector on surface 22_8 is read and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Each sector on the disc pack is preceded by an address field. If the address read before any sector does not match the contents of the disc address registers at that time, both the Error and the Address Error flags are set to 1. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated. The particular type of address error detected can be determined by examining both the Sector Error and the Head Error flags. Three possibilities exist:

1. If both the Sector Error and the Head Error flags are 0, the cylinder number does not match.
2. If the Sector Error flag is 1, the sector number does not match.
3. If the Head Error flag is 1, the head (surface) does not match.

When a new pack is formatted, every sector found to produce irrecoverable errors is tagged with a Bad Sector label. This appears before the address field for the sector. If the drive attempts to read or write in a bad sector, the operation is terminated, and both the Bad Sector flag (which can be read with a READ MEMORY ADDRESS COUNTER instruction) and the Error flag are set to 1. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

If the checkword read at the end of a sector differs from that calculated by the controller, both the Error and the Check Error flags are set to 1. The Read operation is terminated, even if more sectors were supposed to be read; the Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Check Error flag indicates that at least one word in the last sector read contains an error.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The reading of the current sector continues, but once that sector has been read, the Read operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word from the last sector read was not correctly transferred to memory.

If the drive stops tracking on the cylinder, the Seek Error and Error flags are set to 1. The Busy flag is set to 0 and the Done flag is set to 1, thus initiating a program interrupt request.

During Writing

As mentioned above, specifying a non-existent sector will cause the drive unit to search forever for that sector once a Write operation is initiated.

An error can occur when the subsystem is writing a series of sectors in one operation if the Write operation exceeds the number of sectors available. When the last sector on surface 22_8 is written and the drive unit attempts to advance automatically to the next surface, since the sector counter has not overflowed, both the Error and the End Error flags are set to 1. The sector counter is incremented and the sector address is set to sector 0. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated.

Each sector on the disc pack is preceded by an address field. If the address read before any sector does not match the contents of the disc address registers at that time, both the Error and the Address Error flags are set to 1. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated. The particular type of address error detected can be determined by examining both the Sector Error and the Head Error flags. Three possibilities exist:

1. If both the Sector Error and the Head Error flags are 0, the cylinder number does not match.
2. If the Sector Error flag is 1, the sector number does not match.
3. If the Head Error flag is 1, the head (surface) does not match.

When a new pack is formatted, every sector found to produce irrecoverable errors is tagged with a Bad Sector label. This appears before the address field for the sector. If the drive attempts to read or write in a bad sector, the operation is terminated, and both the Bad Sector flag (which can be read with a READ MEMORY ADDRESS COUNTER instruction) and the Error flag are set to 1. The Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

If the data channel does not respond in time to a data channel request, both the Error and the Data Late flags are set to 1. The writing of the current sector continues, but once that sector has been written, the Write operation is terminated. The Busy flag is set to 0, the Done flag is set to 1 and a program interrupt request is initiated. The Data Late flag indicates that at least one word in the sector was not written properly.

If the drive stops tracking on the cylinder, the Seek Error and Error flags are set to 1. The Busy flag is set to 0 and the Done flag is set to 1, thus initiating a program interrupt request.

This page intentionally left blank

SECTION VI

ANALOG / DIGITAL
DIGITAL / ANALOG

- 4120 SERIES A/D CONVERSION SUBSYSTEM
- 4180 SERIES D/A CONVERSION SUBSYSTEM

This page intentionally left blank

INTRODUCTION TO DGC A/D AND D/A SUBSYSTEMS

The DGC Analog Data Conversion System provides a means for interfacing DGC computers to a wide range of analog devices. The Analog/Digital Conversion Subsystem, built around a 10- or 12-bit analog-to-digital (A/D) converter, serves the input function, converting external analog signals into digital data for the central processor. The output function is provided by the Digital/Analog Conversion Subsystem, which uses a 12-bit digital-to-analog (D/A) converter to translate digital data from the computer into analog signal voltages available to external devices.

ANALOG AND DIGITAL DATA

Analog Data

For the purposes of this chapter, analog data consists of voltages whose magnitudes represent, and are proportional to, numeric values. The subsystems may be configured to provide measurements in the following ranges: ±2.5V, ±5V, ±10V, 0 to 5V (A/D only), and 0 to 10V. The A/D and the D/A subsystems may have the same range, or their ranges may be chosen independently.

The total voltage span, either 5, 10, or 20 volts, is known as the "full scale range" (FSR). An analog range is called "unipolar" if it includes only non-negative voltages, "bipolar" if it encompasses both positive and negative voltages. The maximum value within a range is known as "plus full scale" (+FS). For a unipolar range, the minimum value is 0 volts. For a bipolar range, 0 volts in the mid-range value, and the minimum value is called "minus full scale" (-FS).

Digital Data

While analog values (voltages) are continuous over their range, digital data (binary numbers) are discrete--only certain values within the range can be represented. For an n-bit converter these values are the 2^n different n-bit binary integers. They are interpreted in the unipolar case as unsigned integers within the range 0 to 2^n-1, and in the bipolar case, with the most significant bit acting as the sign, as signed integers between -2^{n-1} and $2^{n-1}-1$, inclusive, where two's complement form is used for the negative integers.

Analog/Digital Data Correspondence

In both cases these 2^n numbers define only 2^n-1 intervals, but the analog range is divided into 2^n discrete values, where the difference from one value to the next corresponds to a digital increment equal to the least significant bit (LSB). In this way the "analog LSB", i.e., the voltage increment corresponding to the digital LSB, equals $1/2^n$ of the analog full scale range (FSR), and the analog value represented by each bit position is an exact binary fraction of this full scale range.

The minimum digital value (0 for unipolar, -2^{n-1} for bipolar) corresponds to the minimum analog value (0 or -FS), but the maximum digital value (2^n-1 or $2^{n-1}-1$) actually represents an analog value one analog LSB less than the nominal full scale range. Just as the negative numbers outnumber the (non-zero) positives by one (since 0 is represented, as are the positive numbers, with a 0 sign bit), the actual analog range extends one LSB further in the negative direction than in the positive direction. Thus, the actual maximum analog value (+FS) is either FSR - LSB (unipolar) or 1/2 FSR - LSB (bipolar).

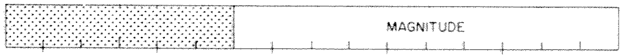
For example, for a 12-bit converter with a ±5 volt range, FSR = 10 volts, and LSB = $\text{FSR}/2^n$ = $10\text{V}/2^{12}$ = .0024 volt. -FS = -5.000 volts, but +FS = 5.000 - .0024 = 4.9976 volts. (The term "full scale" (FS), without a sign prefacing it, usually means the nominal full 5 volts.)

Digital Representation of Analog Data

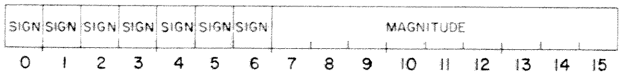
The digital values input or output by a converter consist of either ten or twelve bits of data. This number of bits is referred to as the "resolution" of the converter. When expressing a converter's n-bit digital value in a full 16-bit computer word, the n-bit field of data is right-justified--that is, the least significant data bit occupies the rightmost position, bit 15. Ten-bit values take up bits 6 through 15; 12-bit values occupy bits 4 through 15. Note that this means that the same bit position in two converters with identical ranges but of different resolutions will not represent the same analog voltage level.

When transferring digital values from the computer to a D/A converter, only the rightmost twelve bits of the 16-bit computer word are of importance--the extra high-order bits are ignored and hence may have any value. When reading digital values into the computer from an A/D converter, the extra high-order bits are set by the converter subsystem to fill out the data word according to one of two conventions, one for unipolar ranges, the other for bipolar ones. No sign bit exists in the unipolar case; the extra high-order bits are set to 0. This allows the program to operate on unipolar inputs as if they were normal positive integers. In the bipolar case, the sign of the converter's digital value (the most significant bit in the n-bit field) is extended to the left. In other words, all of the extra high-order bits are set to 0 for a positive converter value and to 1 for a negative value. This convention allows the program to operate on bipolar values as if they were regular signed integers.

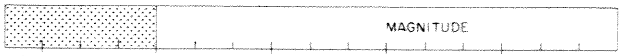
The four possible A/D converter data formats are:



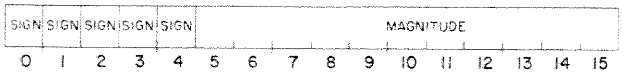
10-bit unipolar



10-bit bipolar



12-bit unipolar



12-bit bipolar

Although the digital values from the A/D converter will necessarily fall within the n-bit range of the converter, normal arithmetic operations on these values can be performed with valid results within the full 16-bit range of the computer. However, if it is desired to keep the results within the converter range (e.g., in order to convert them back to analog form via a D/A converter), care must be taken to check and provide for overflow and underflow out of the narrower range of the converter.

The following table shows the correspondence between analog and digital values for the 10-bit A/D converter and the 12-bit A/D and D/A converters with nominal ranges of 0 to V volts (unipolar) and ±V volts (bipolar). "+V" represents the nominal maximum positive voltages (FS); "-V" represents the minus full scale voltage for the bipolar case (-FS). Inputs to an A/D converter which exceed the range of the converter are clipped (truncated) as follows: any analog input value greater than +FS converts to the maximum digital value; any value less than the minimum analog value (0 or -FS) converts to the minimum digital value. The digital values are given as the octal equivalents of the 16-bit words received from the A/D converter; in D/A conversion the four most significant bits (bits 0 to 3) are ignored and may have any value.

Analog and Digital Data Correspondence

Converter / Analog Voltage	12-bit A/D and D/A		10-bit A/D	
	unipolar	bipolar	unipolar	bipolar
≥+V -LSB	007777	003777	001777	000777
+1/2V	004000	002000	001000	000400
+LSB	000001	000001	000001	000001
0	000000	000000	000000	000000
-LSB		177777		177777
-1/2V		176000		177400
-V +LSB		174001		177001
≤ -V		174000		177000

NOTE: V = FSR (unipolar)
= 1/2FSR (bipolar)

DG-00656

DATA ACCURACY

In converting analog data to digital form, or vice versa, there exist two types of limitation on the accuracy of the conversion; hardware inaccuracies and the conversion process inaccuracies. The first derives from the limitations of the hardware and is severe in proportion to the tolerances and inaccuracies of the components in the system. The measurement of these errors is usually given in terms of "relative accuracy", which is defined as the ratio of the difference between a converter's output (V_{out}) and its given input (V_{in}) to the full scale range:

$$\text{Relative accuracy} = \frac{V_{out} - V_{in}}{FSR} \times 100\%$$

This definition of accuracy incorporates the fact that conversion accuracy is constant over the full scale range, not proportional to the value being measured. The accuracies of the A/D and D/A systems are given separately in their respective chapters.

The second limitation is due to the discrete nature of digital data. Because digital values exist only at certain points, at intervals of one LSB apart from each other, there is an inherent limitation on the accuracy of A/D conversion equal to ±1/2 LSB. In other words, for a given digital output value, the actual analog input to the A/D converter can be anywhere within one half LSB of the analog value specified by the digital output. This "quantization error" must be added to all other analog errors when characterizing the accuracy of an A/D converter subsystem.

On the other hand, except for errors introduced into the D/A converter by actual circuit characteristics and signal noise, the analog output value corresponding to a particular digital input value is exact. However, in spite of appearances, the same ±1/2 LSB error is inherent in D/A conversion but does not appear in the converter; it occurs in the computer when the program rounds or truncates the digital quantities to n bits.

The table below gives the value (in millivolts) of the analog LSB associated with each converter offered with the Analog Data Conversion Subsystem. The inherent inaccuracy of each converter is then plus or minus one half of the value given.

Analog LSB's for Various Converters

Range Resolution	±2.5V	±5V	±10V	0 to 5V	0 to 10V
10 bits	4.88mV	9.77mV	19.53mV	4.88mV	9.77mV
12 bits	1.22mV	2.44mV	4.88mV	1.22mV	2.44mV

DG-00657

This page intentionally left blank

4120 SERIES ANALOG/DIGITAL CONVERSION SUBSYSTEM

SUMMARY

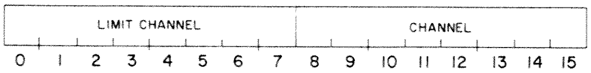
DEVICE MNEMONIC		ADCV
DEVICE CODE		21_8
PRIORITY MASK BIT		8
RESOLUTION	10 bits	12 bits
CONVERSION TIME (MAXIMUM)	13. 3μsec	36μsec
MAXIMUM CONVERSION RATE (TYPICAL)	75kHz	28kHz
SYSTEM ACCURACY		
w/o Prog. Gain	±. 075%±1/2LSB	±.025%±1/2LSB
w/ Prog. Gain	±. 125%±1. 2LSB	±.075%±1/2LSB
APERTURE		40±5ns

ACCUMULATOR FORMATS

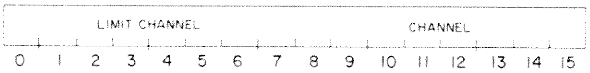
NOTES: [ ] Data Channel Option only.

< > Programmable Gain Option only.

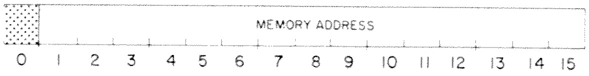
SELECT CHANNEL [AND LIMIT] . . . . DOA



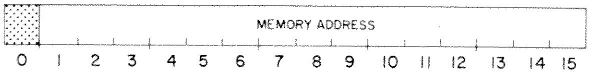
READ CHANNEL SELECT [AND LIMIT] . . DIA



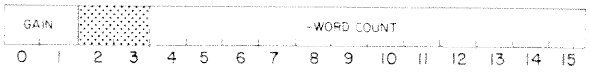
[LOAD MEMORY ADDRESS COUNTER] . . . DOB



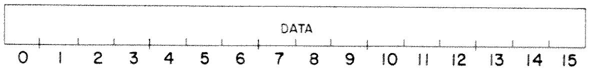
[READ MEMORY ADDRESS COUNTER] . . . DIB



[LOAD WORD COUNT] AND <SELECT GAIN> . . . . DOC



READ DATA . . . . DIC



S, C AND P FUNCTIONS

S	Set Busy to 1, set Done to 0, and either trigger a single conversion or prime the clock.
C	Set Busy and Done to 0.
P	Set Busy to 1, set Done to 0, and trigger a sequence of conversions and data channel transfers or prime the clock.

CONFIGURATIONS

The Analog/Digital Conversion Subsystem enables any DGC computer to read digitized data from external analog measuring devices. Multiplexing of the analog inputs allows a single A/D converter to handle up to 32 analog channels. The subsystem is available with a resolution of either ten or twelve bits and an input range (which is hardware selectable) of ±2.5V, ±5V, ±10V, 0 to 5V, or 0 to 10V. An internal clock, which can be used for synchronizing channel samplings, is standard equipment with all systems. Optional features are the Data Channel Option and the Programmable Gain Option. The Data Channel Option provides automatic sequential or repetitive channel sampling at the maximum possible rate. The digital results are sent automatically to memory via the data channel.

Programmable Gain Option

The Programmable Gain Option, available only on subsystems with an FSR of at least 10 volt, provides for a program-controlled variable gain (voltage multiplication factor) applied to the analog inputs. Gains of 1, 2, 4, and 8 are available and may be selected independently for each conversion or sequence of conversions. If the Data Channel Option is present, the same gain is used for the entire sequence of conversions. This option is useful in a situation where the various analog measuring devices in the system have characteristic ranges of different magnitudes. For example, if the input signals to the A/D converter from at least one device can range up to ±10 volts, the subsystem must be configured with a ±10 volt range to accommodate this device. A device whose signals remain within a ±2.5 volt range would then normally use only one quarter of the total range, which would result in a loss of potential accuracy. However, with the Programmable Gain Option, the original ±10 volt subsystem range may be kept for the first device by assigning it a gain of one, while an effective range of ±2.5 volts may be selected for the second device by choosing a gain of four. Signal voltages from the second device will be multiplied by four before they are converted to digital values. A signal of 2 volts from the second device will then be read as 8 volts and digitized as such. In this way the full ±10 volt subsystem range will be utilized for both devices, thereby maximizing overall subsystem resolution. It is up to the programmer to remember that the digital values representing the analog signals from the second device are actually four times as large as they would be if the Programmable Gain Option were not in use.

The table below shows the effective ranges available with the Programmable Gain Option. This operation is available only on systems with a full scale range of at least 10 volts.

Sub-System Range / Gain	1	2	4	8
±5V	±5V	±2.5V	±1.25V	±.625V
±10V	±10V	±5V	±2.5V	±1.25V
0 to 10V	0 to 10V	0 to 5V	0 to 2.5V	0 to 1.25V

Effective input voltage ranges with Programmable Gain Option

DG-00658

OPERATION

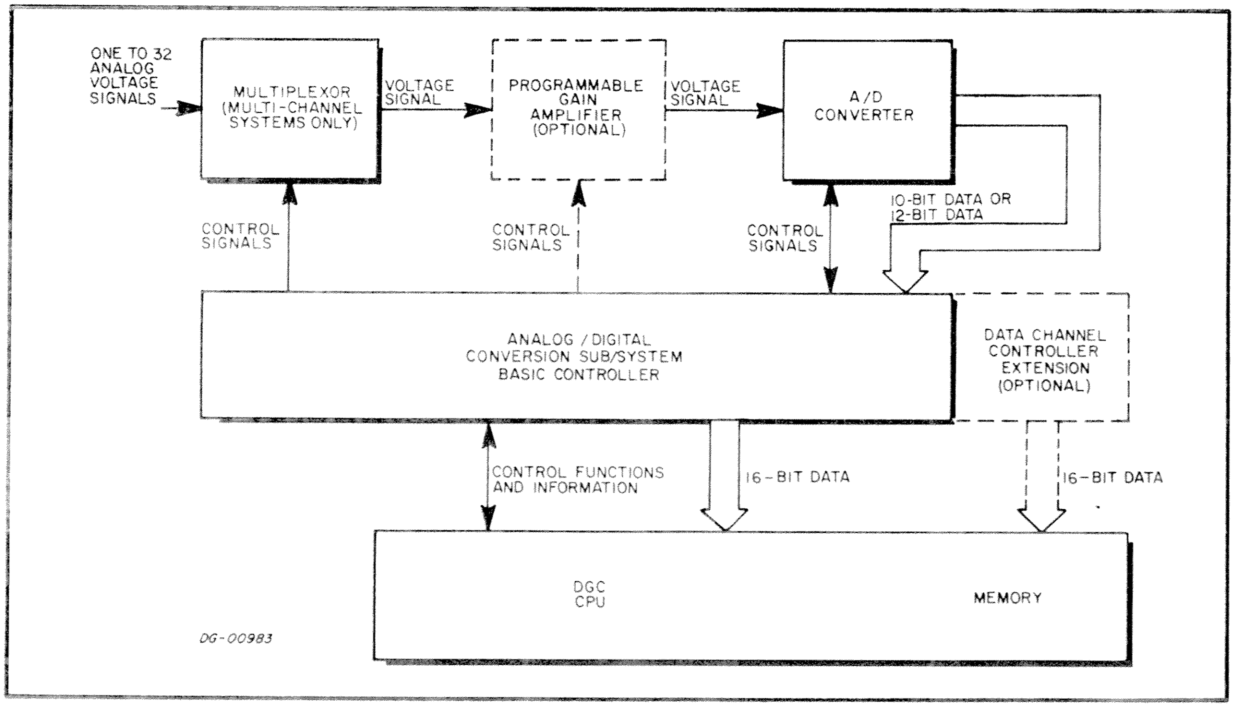
Modes of Operation

There are basically two modes of operation for the A/D Conversion Subsystem. The first allows a program to select a channel, trigger a conversion on that channel, and read the digital result when the conversion is done. This is called "single-conversion mode", since the program must separately initiate each conversion. The second mode allows a program to set up a sequence of conversions and let them proceed automatically, with the digital results being transferred directly to memory via the data channel. This is called "data channel mode" and is available to the program only if the Data Channel Option is present.

The data channel mode is a supplement to, not a replacement for, the operation of the basic subsystem. All of the capabilities of the single-conversion mode remain available to the user with the Data Channel Option. The instruction sets reflect this expansion of capabilities--several of the data channel mode instructions are expanded versions of single-conversion mode instructions. All of the single-conversion mode instructions can still be used, but because these instructions have additional effects when the Data Channel Option is present, special care must be taken when programming in this situation. For this reason, a brief section on single-conversion mode programming with the Data Channel Option present is included in this chapter, following the sections on the two basic modes of programming.

Conversion Sequence

After the program sets up the A/D subsystem as desired, it initiates a conversion by issuing a Start or Pulse command. This sets the Busy flag to 1, the Done flag to 0, and selects either single-conversion mode or data channel mode (S and P commands, respectively). The controller selects the channel specified by a Channel Select Register (previously set up by the program) by sending control signals to the multiplexor. If the system includes the Programmable Gain Option, the input voltage signal selected by the multiplexor is amplified by a factor equal to the selected gain before it enters the A/D converter. The controller then issues a "start conversion" command to the converter. When the conversion is done, the converter sends an end-of-conversion signal back to the controller, indicating that the digital output of the converter is now available to the computer. This digital result, residing in a data register internal to the A/D converter, remains unchanged until a new conversion is begun (or until power is turned off). It may be read repeatedly by the program if desired.


ONE TO 32 ANALOG VOLTAGE SIGNALS
MULTIPLEXOR (MULTI-CHANNEL SYSTEMS ONLY)
VOLTAGE SIGNAL
PROGRAMMABLE GAIN AMPLIFIER (OPTIONAL)
VOLTAGE SIGNAL
A/D CONVERTER
10-BIT DATA OR 12-BIT DATA
CONTROL SIGNALS
CONTROL SIGNALS
CONTROL SIGNALS
ANALOG / DIGITAL CONVERSION SUB/SYSTEM BASIC CONTROLLER
DATA CHANNEL CONTROLLER EXTENSION (OPTIONAL)
CONTROL FUNCTIONS AND INFORMATION
16-BIT DATA
16-BIT DATA
DGC CPU
MEMORY
DG-00983

The effects of the end-of-conversion signal depend on the mode in which the system is operating. In the single-conversion mode, the end-of-conversion signal sets the Busy flag to 0, sets the Done flag to 1, and initiates a program interrupt request. If the data channel mode was selected, the end-of-conversion signal causes a data channel request to be made to the central processor in order to transfer the converter output data directly to memory. As soon as the data channel transfer is complete, the next conversion is begun automatically. When the entire sequence of conversions and data channel transfers has been completed, the Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

The end-of-conversion signal normally has an additional effect in both operating modes--it increments the Channel Select Register in preparation for a conversion on the next sequential channel. This is necessary for automatic sequential channel sampling in the data channel mode and is useful in single-conversion mode when sampling channels sequentially. In some cases, however, it may be desirable not to increment to the next channel automatically. For example, an application may require that individual channels be sampled repeatedly, in either single-conversion or data channel mode. To accommodate these situations, the automatic channel increment feature can be disabled by adjusting the controller.

Synchronization to a Clock

Channel sampling may optionally be synchronized to a clock. A clock is included as standard equipment with all A/D subsystems. The period of this internal clock is mechanically adjustable within the range 10 to 100 microseconds. To provide clock periods outside of this range or a more accurate clock, the user may supply his own external clock.

When the subsystem is configured with either type of clock active, the Start and Pulse commands do not directly trigger a conversion--they prime the clock to do it. In other words, after the Busy flag is set to 1 by the Start or Pulse commands, the next occurring clock pulse will trigger the conversion.

A clock is used in either single-conversion or data channel mode to maintain a constant sampling rate for a sequence of conversions. The first conversion is triggered by the first clock pulse after the Start or Pulse command, and hence may occur at any time up to the clock period. Succeeding conversions will be triggered at the clock frequency, provided that the Busy flag is 1 when each successive clock pulse occurs. In single-conversion mode, the program must issue another Start command (to set the Busy flag back to 1) before each clock period expires in order to maintain the sampling rate. In data channel mode, the Busy flag remains set to 1 until the entire sequence of conversions has been completed, so successive clock pulses will always trigger new conversions. (If, due to an overloaded data channel, the previous converter result has not yet been transferred to memory when the next clock pulse occurs, the previous result is lost and all following conversions in the sequence are invalid. No indication of this is given.)

NOTE At shipping time, the internal clock is disabled. Therefore, unless this clock is enabled or an external one is installed, the Start and Pulse commands will directly trigger conversions.

Power-Up Conditions

When power for the A/D subsystem is turned on, the state of all of the registers in the interface is indeterminate. All relevant registers must therefore be initialized by the program before the first valid conversion can be performed.

The first conversion performed after power-up will always produce a value of minus full scale (-FS); thereafter, conversions will yield correct results. The program can allow for this phenomenon by executing one dummy conversion after power-up. Since the result will be ignored, the program need not initialize any registers for this conversion. An NIOS instruction is therefore sufficient to start the conversion. The program must be sure to wait for this dummy conversion to finish before starting any other conversions.

Channel Numbering

The channels in the system are numbered consecutively from 0 up to one less than the total number of channels. For example, in a 16-channel system the first channel is channel 0 and the last one is channel 15 (channel 17_8).

INSTRUCTIONS: SINGLE-CONVERSION MODE

All of the six available I/O transfer instructions are used by the complete A/D subsystem. They can be grouped into two overlapping instruction sets, one for each basic operating mode. Single-conversion mode programming employs three of these instructions (four if the Programmable Gain Option is present). Five of the six instructions make up the data channel mode instruction set. When programming single conversions in a system which has the Data Channel Option, the expanded data channel mode versions of the three (or four) instructions used in single-conversion mode must be used.

The Busy and Done flags are set to 0 by the Clear command, by the I/O RESET instruction and by the Reset console switch. A Start command in any instruction sets the Busy flag to 1, sets the Done flag to 0, selects single-conversion mode, and triggers a conversion. A Pulse command sets the Busy flag to 1, sets the Done flag to 0, selects data channel mode, and triggers a sequence of conversions.

The single-conversion mode of operation is available on all systems, with or without the Data Channel Option. This section describes the instructions used in single-conversion mode for subsystems which do not include the option. The additional considerations which apply when the system includes data channel operation are given in a later section entitled "Programming: Single-Conversion Mode with Data Channel Option Present".

Basic Controller

The basic A/D subsystem controller contains a Channel Select Register, interrupt control logic, and an internal clock. With the Programmable Gain Option the controller also contains a Gain Select Register.

Channel Select Register

The 8-bit Channel Select Register specifies the channel on which the next conversion is to be performed. It is set up by the program through a SELECT CHANNEL instruction (DOA). It may be read with a READ CHANNEL SELECT instruction (DIA). Unless the automatic channel increment feature is disabled, the Channel Select Register is incremented at the end of each conversion in preparation for a conversion on the next sequential channel.

In a multi-channel subsystem the multiplexor selects the correct channel according to control signals from the Channel Select Register. The subsystem capacity of 32 channels requires only five bits for actual channel selection. The low-order five bits of the Channel Select Register specify the channel; the high-order three bits are ignored at present and are reserved for future system expansion. In a single-channel subsystem there is no multiplexor, and though there still is a Channel Select Register (which can be loaded and read with the SELECT CHANNEL and READ CHANNEL SELECT instructions), it is not meaningful to the controller.

Interrupt Logic and Clock

The controller's interrupt control logic includes the usual Busy and Done flags. The Clear command sets both flags to 0. The Start command sets the Busy flag to 1 and the Done flag to 0. Start triggers a conversion directly if the internal clock is disabled and no external clock is installed. When either type of clock is enabled, the conversion is not triggered until the first clock pulse following the Start command. The end-of-conversion signal from the converter sets the Busy flag to 0, sets the Done flag to 1, and initiates an interrupt request.

Gain Select Register

The controller for a subsystem with the Programmable Gain Option also includes a 2-bit Gain Select Register, which is set up by the SELECT GAIN instruction (DOC). This instruction is used only when the Programmable Gain Option is present. Moreover, when the subsystem includes this option, this instruction must be used, since upon power-up, the Gain Select Register is in an indeterminate state. A SELECT GAIN instruction must be given after power-up and thereafter whenever a gain change is desired.

SELECT CHANNEL

DOA<f> ac,ADCV

0	1	1	AC		0	1	0	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 8-15 of the specified AC are loaded into the Channel Select Register. Bits 0-7 of the specified AC are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

LIMIT CHANNEL								CHANNEL							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-7	---	Reserved for the Data Channel Option.
8-15	Channel	Select channel 0-37_8 for the next conversion.

READ CHANNEL SELECT

DIA<f> ac,ADCV

0	1	1	AC		0	0	1	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Channel Select Register are placed in bits 8-15 of the specified AC. Bits 0-7 of the specified AC are set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

▨								CHANNEL							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	---	Reserved for future use.
8-15	Channel	The next channel to be sampled--in other words, the channel on which the next S (or P) command will trigger a conversion. Unless the automatic channel increment feature is disabled, this channel number is one greater than the number of the channel last sampled.

SELECT GAIN

DOC<f> ac,ADCV

0	1	1	AC		1	1	0	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0 and 1 of the specified AC are loaded into the Gain Select Register. Bits 2-15 of the specified AC are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged.

The SELECT GAIN instruction is used only when the PROGRAMMABLE Gain Option is present. (See Basic Controller section.) If this option is not included in the subsystem, the instruction is a functional "No-op". The format of the specified AC is as follows:

GAIN		▨													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-1	Gain	Select the gain as follows: 00 X1 01 X2 10 X4 11 X8
2-15	---	Reserved for Data Channel Option.

READ DATA

DIC <f> ac,ADCV

0	1	1	AC		1	0	1	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The n-bit output (n=10 or 12) from the A/D converter is placed in the rightmost n bits of the specified AC. If the A/D subsystem is unipolar, the remaining high-order AC bits are set to 0. If the system is bipolar, the remaining high-order AC bits are set to the value of the most significant bit of the converter output (i.e., the sign of the data is extended to the left). After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

10-bit unipolar

▨	▨	▨	▨	▨	▨	MAGNITUDE									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

or

12-bit unipolar

▨	▨	▨	▨	MAGNITUDE											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-5 or 0-3	---	Reserved for future use.
6-15 or 4-15	Magnitude	10- or 12-bit digital representation of the analog value which was converted.

10-bit bipolar

SIGN	SIGN	SIGN	SIGN	SIGN	SIGN	SIGN	MAGNITUDE								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

or

12-bit bipolar

SIGN	SIGN	SIGN	SIGN	SIGN	MAGNITUDE										
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-6 or 0-4	Sign	Sign of the digital representation of the analog value; 0 is positive, 1 is negative.
7-15 or 5-15	Magnitude	9- or 11-bit digital representation of the analog value which was converted. If the sign bits are 1, this is the two's complement of the absolute magnitude of the value.

PROGRAMMING: SINGLE-CONVERSION MODE

The programming for a single-conversion can be divided conceptually into three stages, during which the actions performed by the program are as follows:

1. The controller is set up and the conversion is initiated.
2. The program waits until the conversion is done.
3. The converter data is read into the computer.

The programming for each stage is described more fully below.

Setting up the controller consists of selecting a channel and (optionally) selecting a gain. The SELECT CHANNEL (DOA) and SELECT GAIN (DOC) instructions may be given in either order, but they must both precede the Start command which triggers the conversion. The Start command may be given alone (NIOS), but it is usually appended to the last select instruction given. For consistency it is recommended that the SELECT GAIN instruction (DOC) be given first if it is given at all, followed by the SELECT CHANNEL instruction combined with the Start command (DOAS). (In a single-channel system a DOCS or an NIOS is sufficient.)

To read the converter data into an accumulator the program must give a READ DATA instruction (DIC). This instruction may appear any time after the program has determined that the conversion is done. The converter result accessed by the READ DATA instruction remains available until a new conversion is begun and may be read repeatedly if desired.

The procedures described above, with the following additional note, are sufficient for programming a single isolated conversion. When using interrupts, the Done flag must be set to 0 before exiting from the interrupt handling routine--otherwise, when interrupts are again enabled an unwanted interrupt will occur. The Done flag may be set to 0 by appending to the READ DATA instruction a Clear command (DICC).

To implement a series of conversions, the program should respond to each conversion completion by reading the data and initiating a new conversion. For a series of random channel samplings, the DIC instruction can be followed by a DOAS instruction or a DOC, DOAS sequence. For a series of sequential channel samplings, the program need only issue a Start command to initiate a conversion on the next channel (providing that the automatic channel increment feature is enabled). If the gain need not be switched, the Start command may be appended to the READ DATA instruction (DICS).

Example

The following sample subroutine executes a single-conversion to sample an analog input channel each time it is called. It is called with a JSR CNVRT instruction with the channel number in AC0 and the gain in AC1 (in bits 0 and 1). It returns to the instruction following the JSR CNVRT instruction with the converted data in AC0. The contents of AC3 are destroyed when the JSR is executed. The subroutine does not use interrupts; A/D subsystem interrupts must be disabled.

```
CNVRT: DOC    1,ADCV ;Select gain
       DOAS   0,ADCV ;Select channel and
                     ; start conversion
       SKPDN ADCV    ;Wait for end-of-
       JMP    .-1    ; conversion
       DIC    0,ADCV ;Read converted data
       JMP    0,3    ;Return
```

INSTRUCTIONS: DATA CHANNEL MODE

The Data Channel Option, which consists of an extension to the basic A/D subsystem controller, adds data channel operation to the subsystem. This section describes the instructions used in the data channel mode. For programming single-conversions in a subsystem which has the Data Channel Option, see the following section.

Extended Controller

The extended A/D subsystem controller contains all of the components of the basic controller: the Channel Select Register, the interrupt control logic, and the internal clock. In addition, it contains a Limit Channel Register and the components necessary for data channel operation: a Memory Address Counter, a Word Count Register, and data channel control logic. As with the basic controller, if the subsystem includes the Programmable Gain Option, there is also a Gain Select Register.

Channel Selection Registers

In addition to the 8-bit Channel Select Register, the extended controller contains an 8-bit Limit Channel Register. After a conversion on the channel specified by the Limit Channel Register, the Channel Select Register is set to 0 instead of being incremented. This allows the program to treat the channels from 0 to the limit as a continuous cycle for sequential channel sampling purposes. The SELECT CHANNEL AND LIMIT instruction (DOA) sets up both the Channel Select Register and the Limit Channel Register. It is an extended form of the SELECT CHANNEL instruction used in single-conversion mode. Similarly, the READ CHANNEL SELECT AND LIMIT instruction (DIA), which reads these two registers, is an extended form of the READ CHANNEL SELECT instruction.

Normally, the end-of-conversion signal from the A/D converter increments the Channel Select Register, a comparison is made with the Limit Channel Register, and the Channel Select Register is set to 0 if its incremented contents specify a channel number above the limit. This automatic channel increment feature may be disabled, in which case the Channel Select Register is never incremented--it changes only when set by a SELECT CHANNEL AND LIMIT instruction.

NOTE In this case, however, if the Channel Select Register is set to a channel number greater than or equal to the channel number specified for the Limit Channel Register, the Channel Select Register will be set to 0 after the first conversion on the specified channel, and all subsequent samplings will be on channel 0.

In a multi-channel subsystem the multiplexor selects the correct channel according to control signals from the Channel Select Register. The low-order five bits are sufficient to specify any channel up to the subsystem maximum of 32 channels. The high-order three bits are ignored in actual channel selection and are reserved for future subsystem expansion; however, they are not ignored when the check against the Limit Channel Register is made to determine whether to set the Channel Select Register to 0. In a single-channel subsystem, there is no multiplexor, and though the controller still contains a Channel Select Register and a Limit Channel Register (which can be loaded and read with the SELECT CHANNEL AND LIMIT and READ CHANNEL SELECT AND LIMIT instructions), these registers are not meaningful to the controller.

Interrupt Logic and Clock

As in the basic controller, the interrupt control logic in the extended controller includes the usual Busy and Done flags. The Clear command sets both flags to 0. The Pulse command sets the Busy flag to 1 and the Done flag to 0. The Pulse command directly triggers a sequence of conversions if the internal clock is disabled and no external clock is installed. When either type of clock is enabled, the first conversion is not triggered until the first clock pulse following the Pulse command. Subsequent conversions are triggered on successive clock pulses.

Each end-of-conversion signal from the A/D converter initiates a data channel transfer request to send the converter output to memory. When the entire sequence of conversions has been completed, the Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

Memory Address Counter

The 15-bit Memory Address Counter specifies the address in memory which is to receive, via the data channel, the digital result of the next A/D conversion. It is set up by a LOAD MEMORY ADDRESS COUNTER instruction (DOB) before a sequence of conversions and data channel transfers is begun. After each data channel transfer, the Memory Address Counter is incremented automatically so that successive converter results go into consecutive memory locations.

The Memory Address Counter may be read with a READ MEMORY ADDRESS COUNTER instruction (DIB). This instruction can be used after a complete sequence of conversions to determine the next available memory location--the Memory Address Counter always specifies an address one greater than the address into which the most recent converter data was placed. The instruction may be used at any time, even during a conversion, without disrupting subsystem operation.

Word Counter

The Word Counter is a 12-bit counter used to signal the end of a sequence of conversions in data channel mode. It specifies the two's complement of the number of conversions to be performed in sequence, from 1 up to a maximum of $2^{12} = 4096$ conversions. The Word Counter is automatically incremented along with the Memory Address Counter after each data channel transfer. When the count reaches 0, the Busy flag is set to 0, the Done flag is set to 1, and a program interrupt request is initiated.

The Word Counter is set up by either a LOAD WORD COUNTER instruction or a LOAD WORD COUNTER AND SELECT GAIN instruction, depending upon whether or not the Programmable Gain Option is included in the subsystem. Both are forms of the DOC instruction.

Gain Select Register

The extended controller for a subsystem with the Programmable Gain Option also includes a 2-bit Gain Select Register, identical to that in the basic controller. Subsystems with this option in addition to the Data Channel Option use the same instruction (DOC) to load both the Gain Select Register and the Word Counter. The 2-bit gain and the 12-bit word count are both set up each time the LOAD WORD COUNTER AND SELECT GAIN instruction is used. The desired gain must therefore be coded into the accumulator for this instruction each time the instruction is given, even if it is desired that the gain remain the same. The selected gain is the same for all conversions in a sequence of conversions using the data channel.

SELECT CHANNEL AND LIMIT

DOA <f> ac, ADCV

0	1	1	AC		0	1	0	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 0-7 of the specified AC are loaded into the Limit Channel Register. Bits 8-15 of the specified AC are loaded into the Channel Select Register. A After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

LIMIT CHANNEL								CHANNEL							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	Limit Channel	Maximum channel number for a series of conversions transferred via the data channel.
8-15	Channel	Starting channel for the series of conversions.

READ CHANNEL SELECT AND LIMIT

DIA <f> ac,ADCV

0	1	1	AC		0	0	1	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Limit Channel Register are placed in bits 0-7 of the specified AC. The contents of the Channel Select Register are placed in bits 8-15 of the specified AC. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The format of the specified AC is as follows:

LIMIT CHANNEL								CHANNEL							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-7	Limit Channel	The Limit Channel Register specifies the highest-numbered channel which will be sampled. Following a sampling on the limit channel, the Channel Select Register will be set to 0 instead of being incremented.
8-15	Channel	The Channel Select Register always specifies the next channel to be sampled--in other words, the channel on which the next P (or S) command will trigger a conversion. Unless the automatic channel increment feature is hardware disabled, this channel number is either 0 or one greater than the number of the channel last sampled.

LOAD MEMORY ADDRESS COUNTER

DOB <f> ac,ADCV

0	1	1	AC		1	0	0	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 1-15 of the specified AC are loaded into the Memory Address Counter. Bit 0 of the specified AC is ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

▨	MEMORY ADDRESS														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0	---	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used in a data channel transfer.

READ MEMORY ADDRESS COUNTER

DIB <f> ac,ADCV

0	1	1	AC		0	1	1	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The contents of the Memory Address Counter are placed in bits 1-15 of the specified AC. Bit 0 of the specified AC is set to 0. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F.

The Memory Address Counter always specifies the memory address which is to receive the next data word output by the A/D converter. After a sequence of conversions and data channel transfers has been completed, the Memory Address Counter contains an address one greater than the address into which the last data word was placed. The format of the specified AC is as follows:

▨	MEMORY ADDRESS														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0	---	Reserved for future use.
1-15	Memory Address	Location of the next word in memory to be used in a data channel transfer.

LOAD WORD COUNTER

DOC<f> ac, ADCV

0	1	1	AC		1	1	0	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 4-15 of the specified AC are loaded into the Word Counter. Bits 0-3 of the specified AC are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged.

Bits 4-15 of the specified AC must contain the two's complement of the desired word count. The format of the specified AC is as follows:

▨				-WORD COUNT											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-3	---	Reserved for future use.
4-15	-Word Count	Two's complement of the number of conversions to be made and transferred.

NOTE The LOAD WORD COUNTER instruction is used only when the Programmable Gain Option is not present.

LOAD WORD COUNTER AND SELECT GAIN

DOC<f> ac, ADCV

0	1	1	AC		1	1	0	F		0	1	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits 4-15 of the specified AC are loaded into the Word Counter. Bits 0 and 1 of the specified AC are loaded into the Gain Select Register. Bits 2 and 3 of the specified AC are ignored. After the data transfer, the controller's Busy and Done flags are set according to the function specified by F. The contents of the specified AC remain unchanged.

Bits 4-15 of the specified AC must contain the two's complement of the desired word count. The format of the specified AC is as follows:

GAIN		▨		-WORD COUNT											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents
0-1	Gain	Select the gain for the sequence of conversions as follows: 00 X1 01 X2 10 X4 11 X8
2-3	---	Reserved for future use.
4-15	-Word Count	Two's complement of the number of conversions to be made and transferred.

NOTE The LOAD WORD COUNTER AND SELECT GAIN instruction is used when the Programmable Gain Option is present.

Programming

The programming for data channel operation of the A/D subsystem comprises two stages:

1. The controller is set up and the sequence of conversions is initiated.
2. The program waits until the sequence of conversions is complete before accessing the converter results in memory.

No programming is necessary for reading the results into the computer, since the converter data is transferred automatically to memory via the data channel after each conversion. For this reason, the data channel mode provides rapid, automatic sequencing of conversions with minimal supervision from the program.

Setting up the extended controller consists of loading up to five registers, including the optional Gain Select Register. The three output instructions--SELECT CHANNEL AND LIMIT (DOA), LOAD MEMORY ADDRESS COUNTER (DOB), and either LOAD WORD COUNTER or LOAD WORD COUNTER AND SELECT GAIN (DOC)--accomplish this initialization. (In a single-channel subsystem the SELECT CHANNEL AND LIMIT instruction is unnecessary.) These instructions may be given in any order, but they must all precede the Pulse command. The Pulse command may be given alone (NIOP), but it is usually appended to the last select instruction given.

When all conversions in the sequence have been completed, the Done flag will be set to 1. At this point the results for the entire sequence of conversions are in memory and accessible to the program. The program can diagnose this condition in the usual ways, by testing the Done flag with an I/O SKIP instruction or by allowing a program interrupt to occur.

Initial word count values are loaded into the Word Counter from only the rightmost twelve bits of an accumulator. If the Programmable Gain Option is not present in the subsystem, accumulator bits 0-3 are ignored when the LOAD WORD COUNTER instruction (DOC) is given. The program may, therefore, load the accumulator with the normal 16-bit representation of the word count (between -1 and -4096) instead of the less convenient 12-bit representation. However, if the subsystem includes the Programmable Gain Option, bits 0 and 1 of the LOAD WORD COUNTER AND SELECT GAIN instruction (DOC) are used to select the gain. In this case the desired gain must be coded into accumulator bits 0 and 1 every time the word count is initialized.

Example

The following sample programming excerpt causes the extended controller to execute a sequence of conversions on channels 5 through 7 and on 0 through 2, in that order. The gain selected is two. The results are put into six locations in memory, beginning at BUFF. This code may be used with or without interrupts.

```
          .
          .
          .
          LDA   0,CHANS
          DOA   0,ADCV     ;Select channel and
                           ; limit
          LDA   0,ABUFF
          DOB   0,ADCV     ;Load Memory Ad-
                           ; dress Counter
          LDA   0,GWDCT
          DOCP  0,ADCV     ;Load word count and
                           ; gain and trigger se-
                           ; quence of conver-
                           ; sions
          .
          .
          .
CHANS:    3405             ;Limit channel of 7,
                           ; channel select of 5
ABUFF:    BUFF             ;Target location
GWDCT:    47772            ;Gain of 2, word count
                           ; of -6
BUFF:     .BLK  6          ;6-word block for data
          .
          .
          .
```

PROGRAMMING: SINGLE-CONVERSION MODE WITH DATA CHANNEL OPTION PRESENT

The programming of single-conversions in a subsystem that includes the Data Channel Option is a cross between the programming of the two basic modes. The methods are those of the single-conversion mode with the addition of the limit channel capability of the data channel mode. The instruction sequences are the same as in single-conversion mode, but since the extended controller is present, the data channel mode versions of some of these instructions must be employed.

In single-conversion mode the SELECT CHANNEL instruction (DOA) loads the Channel Select Register with the channel number contained in bits 8-15 of the specified accumulator; bits 0-7 of this AC are ignored and hence are usually programmed (by default) as zeroes. When the Data Channel Option is present, the same DOA instruction also sets up the Limit Channel Register, according to AC bits 0-7. If the program were to use the normal single-conversion mode channel select word, the Limit Channel Register would be set to 0. At the end of the conversion following the SELECT CHANNEL instruction, the Channel Select Register would be set to 0 instead of being incremented and automatic sequential channel selection would be disrupted. The normal sequential channel selection of single-conversion mode can be preserved by insuring that the Limit Channel Register always specifies a higher-numbered channel than the Channel Select Register does. The simplest way to do this is to set the Limit Channel Register to its maximum value, namely 255 (377_8), in which case sequences of single-conversions can be programmed just as if the Data Channel Option were not present.

By setting the Limit Channel Register to its maximum value, its limit channel function is effectively disabled. The program also has the option, however, of using this limit channel capability by setting the Limit Channel Register to some smaller value. In this case the channels from 0 to the limit are treated as a continuous cycle, just as in data channel mode. This capability would be useful if the program wanted to sample sequential channels cyclically, but not at regular intervals as in data channel mode.

A modified version of the single-conversion mode sample program CNVRT could be used to implement the sampling sequence described above. After setting up the gain (if necessary) and selecting the first channel and the limit channel, the following subroutine would be called (with a JSR NEXTC instruction) to trigger a conversion on the next sequential channel and return with the result in AC0. (A/D interrupts must be disabled, as before.)

```
NEXTC: NIOS     ADCV      ;Trigger next conver-
                          ; sion
       SKPDN    ADCV      ;Wait until done
       JMP      .-1
       DIC      0,ADCV    ;Read data
       JMP      0,3       ;Return
```

Except for the limit channel effects described above, single-conversion mode programming is the same for systems with and without the Data Channel Option.

TIMING CONSIDERATIONS

Maximum Conversion Time

Conversion time is defined as the time between the Start or Pulse command and the moment when either an interrupt or a data channel transfer is requested (in single-conversion mode and data channel mode, respectively). Maximum conversion time is the maximum amount of time that a conversion could require in an A/D subsystem of a given resolution. A given converter in a given controller will always take the same amount of time to perform its conversions. This actual conversion time depends on the given converter and controller hardware and is variable to a certain extent among different subsystems. For this reason maximum conversion times are specified for the A/D subsystem. These times are 13.3μsec for 10-bit subsystems and 36μsec for 12-bit subsystems. Subsystems of a given resolution are guaranteed to have their conversion times less than or equal to the maximum specified for that resolution.

Typical Maximum Conversion Rate

The maximum possible conversion rate is attained by a given subsystem when it is operating in data channel mode without a clock enabled. (If there are other peripherals also using the data channel, the A/D subsystem must be given the highest priority for this to be true.) The maximum conversion rate of a subsystem depends not only on the fixed conversion time, but also on the speed of the data channel in the computer being used. For this reason, typical maximum conversion rates are specified. For the 10-bit and 12-bit subsystems these typical maximum conversion rates are roughly 75,000 and 28,000 conversions per second, respectively.

Clock Settings

As described earlier, the period of the internal clock can be adjusted between 10 and 100 microseconds (yielding frequencies of between 10kHz and 100kHz), and an external clock may be installed for more accuracy or for periods outside this range. No matter what clock range is available, however, there is an operational lower limit on the clock period which is determined by the configuration of the given subsystem. The clock period must be greater than the conversion time, or else clock pulses will start new conversions before the current ones are finished, producing unpredictable results. In fact, the clock period must be enough greater than the conversion time to allow for either the maximum total interrupt latency in single-conversion mode or the maximum data channel latency in data channel mode.

Additional Considerations

The program should be careful not to start a new conversion while one is still in progress, as this produces unpredictable results. More generally, during a conversion or sequence of conversions, the program should normally avoid giving any instructions which alter the states of any of the registers or flags in the controller.

The Clear command is normally used only to set the Done flag to 0 after an interrupt has occurred if another conversion is not to be started immediately. However, if it is necessary to idle the A/D subsystem while it is running, the Clear command can also be used for this purpose. It should be noted that, although the Clear command idles the subsystem controller, it does not abort a conversion already begun by the A/D converter. Once an actual conversion has been started, it will necessarily proceed to completion. Setting the Busy flag to 0 with the Clear command merely inhibits the initiation of either the interrupt request or the data channel transfer request which normally would have occurred when the conversion was completed. Thus, if a Clear command is given during a conversion, the subsystem is not really idle until the conversion has finished. For this reason, the program should wait at least one conversion time after idling a running subsystem before attempting to start a new conversion or sequence of conversions.

In single-conversion mode the Clear command might be used to ignore the conversion in progress if the subsystem were then to remain idle for some time. However, there is little reason to use the Clear command when executing a single-conversion if it is desired to start a new conversion right away. Even after giving the Clear command, the program would have to wait for the conversion to finish before starting another. For this reason it is more efficient for the program to allow the end-of-conversion interrupt ot occur, to ignore the conversion by not reading in the result, and to start the new conversion immediately.

When used in data channel mode the Clear command causes termination of the sequence of conversions being performed after the conversion in progress is completed. The result of this conversion will not be transferred to memory, and no further conversions of the sequence will be performed. (The results of all conversions completed prior to the Clear command will have been transferred properly to memory provided the data channel responded in time to each request.) The Word Counter will contain the negative of the number of conversions by which the sequence was shortened, and the Memory Address Counter will contain the address into which the result from the conversion in progress would have gone. Again, the program must allow enough time for the actual conversion in progress to finish before starting a new conversion or sequence of conversions.

In data channel mode, as in single-conversion mode, there is a more efficient way than the Clear command to terminate the current sequence of conversions when another is to be started right away. The program need only set the word count to -1 while the sequence of conversions is being executed. When the conversion in progress finishes, the result will be transferred to memory as usual. The Word Counter will be incremented up to 0, thereby terminating the sequence of conversions and requesting an interrupt. The program can then respond to the interrupt by starting a new conversion or sequence of conversions immediately.

4180 SERIES DIGITAL/ANALOG CONVERSION SUBSYSTEM

CONFIGURATIONS

The Digital/Analog Conversion subsystem provides any DGC computer with the capability of directly controlling external analog devices. The subsystem may be configured with either one or two analog output channels. The output voltage range may be mechanically selected independently for each channel from among the following ranges: ±2.5V, ±5V, ±10V, and 0 to 10V. (Note: Unipolar and bipolar ranges may not be mixed in the same subsystem.) The subsystem resolution is twelve bits, providing 4096 discrete analog values, regardless of the range or ranges chosen.

Scope Control Option

The Scope Control Option provides all necessary control functions for interfacing the computer to any typical storage or non-storage oscilloscope. Two output channels are required for this application--one to drive the horizontal input and one for the vertical input of ths oscilloscope. The operation and programming of the Scope Control Option are fully described later in this chapter.

OPERATION

The subsystem controller contains a separate 12-bit data register and D/A converter for each output channel. Each D/A converter continuously converts the digital value stored in its corresponding data register to an analog output signal. Whenever the contents of a data register are changed by the program, the converter on that channel begins converting the new value immediately. It takes five microseconds before the new digital value is accurately represented by the analog output value. The output remains constant at the new analog value until the data register's contents are changed again.

The controller also contains a Channel Select Bit to specify the channel. The two channels are numbered 0 and 1, corresponding to the value of this bit. When digital data is sent to the controller for conversion, the Channel Select Bit determines which data register will receive the new value.

SUMMARY

DEVICE MNEMONIC .... DACV

DEVICE CODE .... 23_8

PRIORITY MASK BIT.................. None

RESOLUTION .... 12 bits

SETTLING TIME TO 0.01%
OF FINAL VALUE................. 5μsec

ACCURACY .... ±0.02% of FSR

ACCUMULATOR FORMATS

SELECT CHANNEL .... DOB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CHANNEL

OUTPUT DATA AND CONVERT ....DOA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	DATA											

SCOPE CONTROL

SELECT SCOPE MODE................. DOC

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MODE	

READ ERASE STATUS .... DIA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	ERASE STATUS

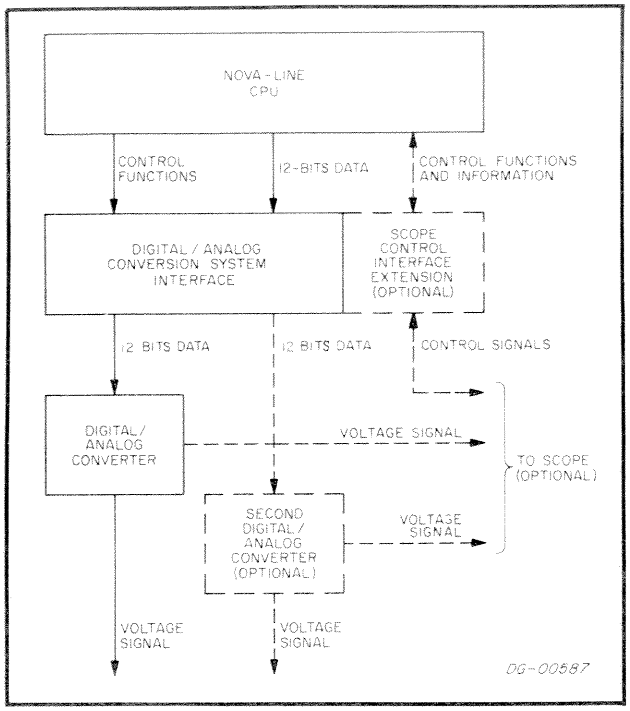
S, C AND P FUNCTIONS

(SCOPE CONTROL ONLY)

S	Z-axis control
C	No effect
P	Erase scope

When power is turned on for the subsystem, the data registers and the Channel Select Bit are in indeterminate states. The initial outputs of the D/A converters are hence undefined.

The Channel Select Bit is present even in a single-channel subsystem. To insure that this single-channel (channel 0) is operable, channel 0 must be explicitly selected (once) after each power-up.


NOVA - LINE CPU
CONTROL FUNCTIONS
12-BITS DATA
CONTROL FUNCTIONS AND INFORMATION
DIGITAL / ANALOG CONVERSION SYSTEM INTERFACE
SCOPE CONTROL INTERFACE EXTENSION (OPTIONAL)
12 BITS DATA
12 BITS DATA
CONTROL SIGNALS
DIGITAL/ ANALOG CONVERTER
VOLTAGE SIGNAL
TO SCOPE (OPTIONAL)
SECOND DIGITAL/ ANALOG CONVERTER (OPTIONAL)
VOLTAGE SIGNAL
VOLTAGE SIGNAL
VOLTAGE SIGNAL
DG-00587

The D/A subsystem controller has no data channel or interrupt facilities. Interrupt facilities are not required since there is no need for the program to know when a conversion is complete. The program need not wait for the end of a conversion before proceeding with another, nor must it perform any specific action when a conversion is complete. Similarly, flag control commands are unnecessary--the D/A converters are always converting whatever values are contained in the data registers, and new values are converted automatically when they are placed in the registers. (The Start and Pulse commands are used by the Scope Control Option; the Clear command has no effect.)

INSTRUCTIONS: D/A

The basic D/A conversion subsystem (without the Scope Control Option) uses only two I/O instructions--one selects the channel and the other sends digital data to the controller. Interrupts and the flag control commands (Start, Clear, and Pulse) are not used by the basic controller.

SELECT CHANNEL

DOB <f> ac, DACV

0	1	1	AC	1	0	0	F	0	1	0	0	1	1
0	1	2	3 4	5	6	7	8 9	10	11	12	13	14	15

Bit 15 of the specified AC is loaded into the Channel Select Bit. Bits 0-14 of the specified AC are ignored. After the data transfer, the function specified by F is generated. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

▨	CHANNEL
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14	15

Bits	Name	Contents
0-14	---	Reserved for future use.
15	Channel	Select the channel 0 or 1.

OUTPUT DATA AND CONVERT

DOA <f> ac, DACV

0	1	1	AC	0	1	0	F	0	1	0	0	1	1
0	1	2	3 4	5	6	7	8 9	10	11	12	13	14	15

Bits 4-15 of the specified AC are loaded into the data register for the channel specified by the Channel Select Bit. Bits 0-3 of the specified AC are ignored. After the data transfer, the function specified by F is generated. The contents of the specified AC remain unchanged. The two allowable formats of the specified AC are as follows:

▨	MAGNITUDE
0 1 2 3	4 5 6 7 8 9 10 11 12 13 14 15

Unipolar systems

▨	SIGN	MAGNITUDE
0 1 2 3	4	5 6 7 8 9 10 11 12 13 14 15

Bipolar systems

Bits	Name	Contents
0-3	---	Reserved for future use.
4 (bipolar)	Sign	Sign bit for magnitude 0 = positive, 1 = negative.
4-15 (unipolar) 5-15 (bipolar)	Magnitude	Digital value to be converted to an analog value.

PROGRAMMING: D/A

To generate a single-conversion, the program can give a SELECT CHANNEL instruction (DOB) followed by an OUTPUT DATA AND CONVERT instruction (DOA). If consecutive conversions are on the same channel, it is not necessary to reselect the channel. For a sequence of conversions on the same channel, the program need issue only one SELECT CHANNEL instruction followed by any number of OUTPUT DATA AND CONVERT instructions.

For a sequence of conversions on alternate channels, the program may simplify the channel selection by merely incrementing between conversions the accumulator used to specify the channel in the SELECT CHANNEL instruction. Only bit 15 of this accumulator participates in channel selection; the other bits are ignored. Of course, the accumulator's contents must not be destroyed between channel selections.

The fact that the high-order four bits of data words are ignored when the OUTPUT DATA AND CONVERT instruction is given simplifies both the initialization of negative data words in an assembly language program and the arithmetic operations on signed data during program execution. The program may set up and use normal signed as well as unsigned integers with correct results as long as the actual 12-bit range of the system is not exceeded. For unipolar systems the actual range is 0 to 4095 (0 to 7777_8); for bipolar systems it is -2048 to +2047 (174000_8 to 3777_8).

Timing and Accuracy Considerations

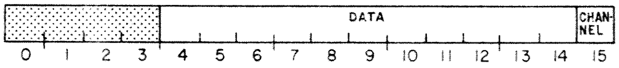
The contents of a data register are changed when an OUTPUT DATA AND CONVERT instruction is given. However, it takes a significant amount of time for the D/A converter to switch its analog output value accordingly. The value that the analog output ultimately attains, corresponding to the new digital value, is known as the "final value". This final value represents the new digital value to an accuracy of 0.02% of the full scale range.

After the conversion of a new digital value is begun, analog output value is indeterminate for up to five microseconds. At the end of this 5μsec "settling time", the analog output value is guaranteed to be within 0.01% of the final value. Note that the converter output may be in error as much as 0.02% + 0.01% = 0.03% at this point. If no new digital data is introduced to the data register, the converter output will become accurate to within the specified 0.02% a short time later.

Despite the settling time described above, there is no harm in starting another conversion on a channel within five microseconds of the beginning of the last conversion on that channel. The converter is always converting the contents of its data register, so the aborted conversion is "forgotten" and the conversion of the new digital value is begun immediately.

Example

The following sample subroutine DASEQ performs a rapid sequence of D/A conversions. It is entered by executing a JSR DASEQ instruction in the calling program. AC1 contains the two's complement of the number of conversions to perform; AC2 contains the address of a block of data. Each word of data in the block is presumed to contain twelve bits of converter data in bits 3-14 and the desired channel number in bit 15 as follows:



The subroutine fetches each data word in turn, selects the channel according to bit 15, right-justifies the 12-bit data, outputs the data to the selected channel, and increments the word count. When the count reaches 0, the subroutine returns to the instruction in the calling program following the JSR DASEQ instruction.

```
DASEQ: LDA   0,0,2     ;Fetch data to AC0
       DOB   0,DACV    ;Select channel
       MOVR  0,0       ;Shift 12-bit data into
                       ; place
       DOA   0,DACV    ;Output data and convert
       INC   2,2       ;Point to next data word
       INC   1,1,SZR   ;Increment word count
       JMP   DASEQ     ;Loop back
       JMP   0,3       ;Return
```

SCOPE CONTROL OPTION

The Scope Control Option provides for program control of any typical storage or non-storage oscilloscope. Capabilities include display screen operation, beam intensification control, storage scope erasing, and two additional signals that may be used as needed to control special aspects of a particular scope.

Operation

Display Screen Operation

Operating an oscilloscope as a display screen requires that the scope have an external horizontal input and that the D/A conversion system include two output channels. The two channels are connected to the horizontal and vertical inputs (X and Y axes) of the scope. Individual points are located on the screen by specifying their X and Y coordinates.

Since the digital range of each D/A channel encompasses $2^{12} = 4096$ different values, the scope screen appears to the program as a 4096 x 4096 dot matrix. This is generally more than enough points to enable the illusion of continuous lines on the screen. To specify a point the program supplies the two coordinates by setting up both output channels with the correct values. The usual convention is to connect channel 0 to the X input and channel 1 to the Y input, but this is by no means mandatory.

Intensification Control

The intensity of the scope beam (known as the "Z-axis" of the scope) can be controlled by a signal which is sent to the scope when the Start command is generated by the program. This intensity signal is a pulse, with a beginning, duration, and end. The characteristics of this Z-axis pulse can be adjusted mechanically to meet the requirements of the scope and the programming.

The duration of the Z-axis pulse, as well as the delay between the issuance of the Start command and the beginning of the pulse, can be adjusted within the range 1.4 to 6 microseconds. The amplitude of the Z-axis pulse is variable in several increments, and both ac and dc coupling are available.

The Z-axis pulse may be a "blanking" or "unblanking" pulse. A blanking pulse makes the beam disappear for the duration of the pulse, with normal intensity returning at the end of the pulse. An unblanking pulse turns up the intensity to a level adequate for viewing for the pulse's duration and returns the screen to its blank state when the pulse is terminated.

The scope may be operated without the Z-axis control, keeping the beam always at normal intensity. In this case, however, "ghost" lines appear when the beam is moved from one point to another since the beam sweeps across the screen. One purpose of the Z-axis control is to eliminate these lines. The blanking type of pulse can be used to turn off the beam temporarily while the beam's coordinates are being changed. When the unblanking type of pulse is used, on the other hand, the screen is kept blank except for short periods of intensification corresponding to the durations of the pulses. The unblanking type of pulse is more commonly used since the programming is simpler--the program merely intensifies the beam temporarily with a Start command to display each point.

NOTE When using either a blanking Z-axis pulse or no Z-axis control at all, be careful not to harm the scope screen by leaving the beam intensified for an extended period of time without changing the coordinates.

Erase Control

The stored image of a storage scope may be erased under program control by issuing a Pulse command. Erasing can also be accomplished by means of a manual switch on the scope's front panel. Erase time is typically in the range one-tenth to one-half second.

If the particular scope being used generates an "erase status" signal which is properly connected to the D/A subsystem, the program can determine whether the scope is presently being erased by giving a READ ERASE STATUS instruction (DIA).

Additional Control Signals

Two additional control signals are provided which may be used to select one of several scope operating modes under program control. These two signals are independent of each other, and each is two-valued. The two possible values for each can be thought of as binary 0 and 1 or as logical false and true. These signals act like flags in that they remain in one state until the program assigns them a different value. The SELECT SCOPE MODE instruction (DOC) sets the value of both of these signals at once.

The use, if any, of these control signals is determined by the requirements and features of the particular scope being used. In one typical usage with a storage scope, one of the signals selects storage versus non-storage mode while the other selects "write-through" mode when the scope is in storage mode. In write-through mode points are displayed on the scope without being stored, while the information already stored is not affected.

INSTRUCTIONS: SCOPE CONTROL

The Scope Control Option uses the two instructions described for the basic D/A subsystem, two additional instructions described below, and the Start and Pulse commands. As described previously, a Pulse command erases the stored image of a storage scope, and a Start command is used to control the beam intensity. The Clear command is not used (it has no effect on the D/A controller), and there are no interrupt or data channel facilities.

SELECT SCOPE MODE

DOC <f> ac, DACV

0	1	1	AC		1	1	0	F		0	1	0	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The scope mode is selected according to bits 14 and 15 of the specified AC. The effect of these bits depends on the scope and how it is interfaced to the D/A conversion sybsystem. Bits 0-13 of the specified AC are ignored. After the mode selection, the function specified by F is performed. The contents of the specified AC remain unchanged. The format of the specified AC is as follows:

▨														MODE	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Function
0-13	---	Reserved for future use.
14-15	Mode	Depends on the particular scope used and the manner in which it is interfaced to the controller.

READ ERASE STATUS

DIA <f> ac, DACV

0	1	1	AC		0	0	1	F		0	1	0	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The scope erase status is read into bit 15 of the specified AC. Bits 0-14 of the specified AC are set to 0. After the data transfer, the function specified by F is performed. AC is formatted as follows:

▨															ERASE STATUS
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Meaning if 1
0-14	---	Reserved for future use.
15	Erase Status	Scope is presently being erased.

PROGRAMMING: SCOPE CONTROL

To display a point on the scope screen, the program must supply the X and Y coordinates and then intensify the beam. To supply each coordinate, the program must give a SELECT CHANNEL instruction (DOB) to select the correct axis followed by an OUTPUT DATA AND CONVERT instruction (DOA) to specify the coordinate itself. The method for intensifying the beam depends on the type of Z-axis pulse for which the hardware is configured. For the unblanking Z-axis pulse, the program need only generate a single Start command by appending an "S" mnemonic modifier to the last I/O instruction which specifies the coordinates. The example program at the end of this chapter illustrates the use of the unblanking Z-axis pulse.

If the Z-axis pulse is a blanking pulse, it is generally used to blank the screen during the time it takes to switch the coordinates of the beam from one point to the next. In this case, three things are necessary:

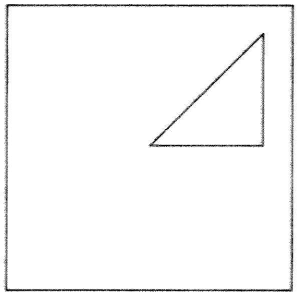
1. The SELECT CHANNEL and OUTPUT DATA AND CONVERT instructions required to move the scope beam to the next point should follow each other in close succession;
2. The Start command should be included with each of these I/O instructions to keep the screen blank until all coordinate switching is done; and
3. The delay of the Z-axis pulse should be adjusted to its minimum value and the duration of the Z-axis pulse should be near its maximum value (at least 5μsec).

For example, with the (blanking) Z-axis pulse delay set at 1.4μsec and the duration set at 5μsec, the following code moves the scope beam from its current position to the point whose coordinates are stored at XCO and YCO, blanking the screen while the coordinates are being changed.

```
LDA   1,XCO    ;AC1 holds X coordinate
SUB   0,0      ;Set AC0 to 0
DOBS  0,DACV   ;Select X axis and blank
               ; screen
DOAS  1,DACV   ;Set new X coordinate and
               ; continue blanking
INC   0,0      ;Set AC0 to 1
DOBS  0,DACV   ;Select Y axis and continue
               ; blanking
LDA   1,YCO    ;AC1 holds Y coordinate
DOAS  1,DACV   ;Set new Y coordinate and
               ; continue blanking during
               ; settling time
```

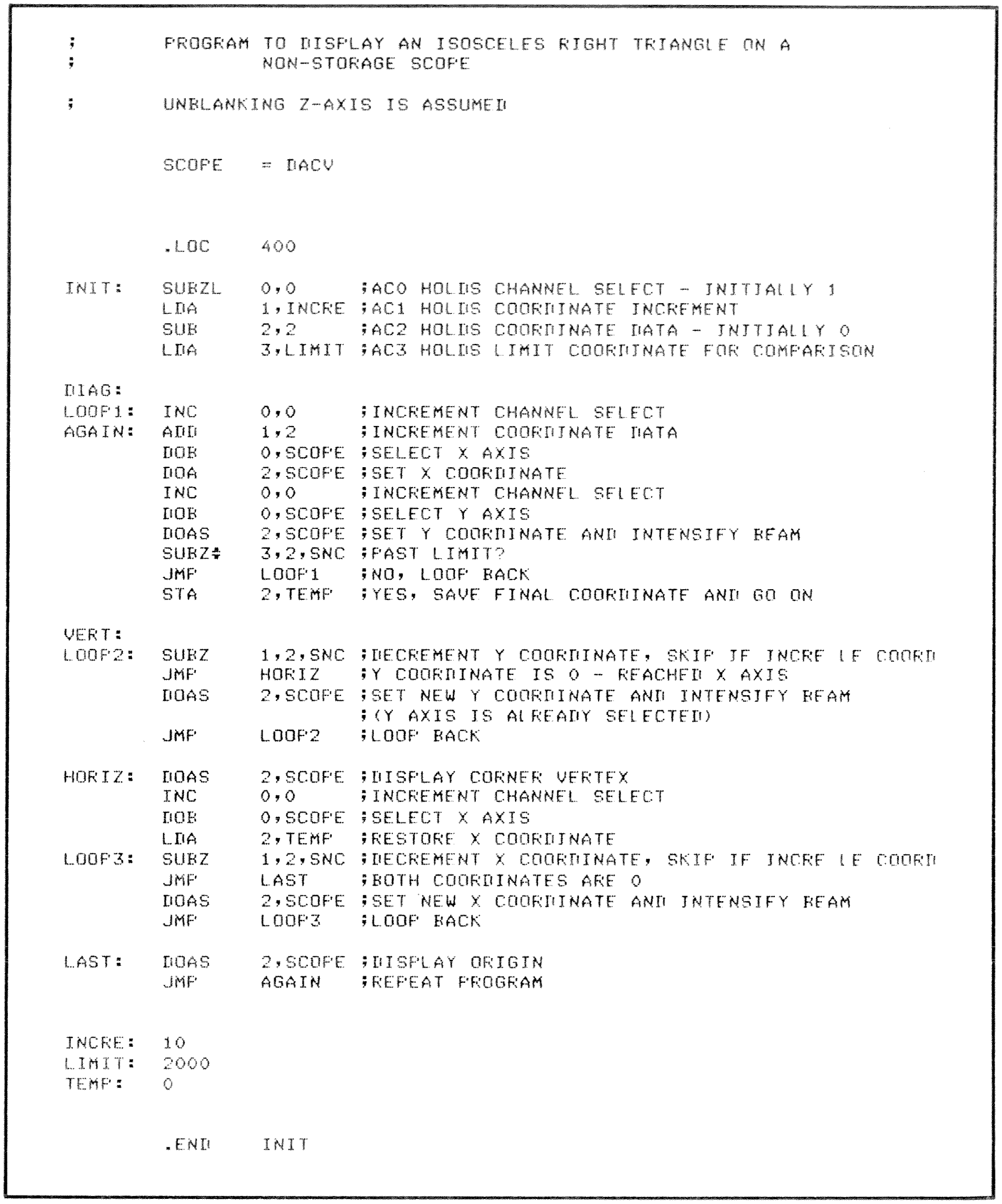
Example

The following example program produces the image of a right isosceles triangle on a typical non-storage oscilloscope. The Z-axis pulse is assumed to be of the unblanking type. The resulting display appears as follows on the scope screen:



The program first initializes the accumulators. AC0 is used with the SELECT CHANNEL instructions to select the axes. AC1 holds, throughout the program, an incremental coordinate value (INCRE) which is added to or subtracted from the coordinates of the current point to produce the coordinates of the next point. AC2 contains current coordinate data, either for the X axis (horizontal leg of the triangle), the Y axis (vertical leg), or both (hypotenuse). AC3 always contains a limiting coordinate value (LIMIT); comparisons between this value and the current coordinate value in AC2 enable the program to tell when it has reached the upper vertex at the end of the diagonal leg.

The main loop of the program, which is repeated indefinitely, comprises three smaller loops which "draw" the three sides of the triangle. The program starts at the origin and draws the hypotenuse at a 45° angle by setting the X and Y coordinates equal for each point. When the limiting value is reached, the program draws down the vertical side by outputting successively smaller Y coordinates without changing the X coordinate. When the Y coordinate becomes 0, the program displays the corner vertex and uses a similar procedure to draw the horizontal base. This time, when the X coordinate becomes 0, the program displays the origin (not displayed as part of any leg) and starts again on the hypotenuse.



```
;       PROGRAM TO DISPLAY AN ISOSCELES RIGHT TRIANGLE ON A
;               NON-STORAGE SCOPE

;       UNBLANKING Z-AXIS IS ASSUMED


        SCOPE   = DACV



        .LOC    400

INIT:   SUBZL   0,0     ;AC0 HOLDS CHANNEL SELECT - INITIALLY 1
        LDA     1,INCRE ;AC1 HOLDS COORDINATE INCREMENT
        SUB     2,2     ;AC2 HOLDS COORDINATE DATA - INITIALLY 0
        LDA     3,LIMIT ;AC3 HOLDS LIMIT COORDINATE FOR COMPARISON

DIAG:
LOOP1:  INC     0,0     ;INCREMENT CHANNEL SELECT
AGAIN:  ADD     1,2     ;INCREMENT COORDINATE DATA
        DOB     0,SCOPE ;SELECT X AXIS
        DOA     2,SCOPE ;SET X COORDINATE
        INC     0,0     ;INCREMENT CHANNEL SELECT
        DOB     0,SCOPE ;SELECT Y AXIS
        DOAS    2,SCOPE ;SET Y COORDINATE AND INTENSIFY BEAM
        SUBZ#   3,2,SNC ;PAST LIMIT?
        JMP     LOOP1   ;NO, LOOP BACK
        STA     2,TEMP  ;YES, SAVE FINAL COORDINATE AND GO ON

VERT:
LOOP2:  SUBZ    1,2,SNC ;DECREMENT Y COORDINATE, SKIP IF INCRE LE COORD
        JMP     HORIZ   ;Y COORDINATE IS 0 - REACHED X AXIS
        DOAS    2,SCOPE ;SET NEW Y COORDINATE AND INTENSIFY BEAM
                        ;(Y AXIS IS ALREADY SELECTED)
        JMP     LOOP2   ;LOOP BACK

HORIZ:  DOAS    2,SCOPE ;DISPLAY CORNER VERTEX
        INC     0,0     ;INCREMENT CHANNEL SELECT
        DOB     0,SCOPE ;SELECT X AXIS
        LDA     2,TEMP  ;RESTORE X COORDINATE
LOOP3:  SUBZ    1,2,SNC ;DECREMENT X COORDINATE, SKIP IF INCRE LE COORD
        JMP     LAST    ;BOTH COORDINATES ARE 0
        DOAS    2,SCOPE ;SET NEW X COORDINATE AND INTENSIFY BEAM
        JMP     LOOP3   ;LOOP BACK

LAST:   DOAS    2,SCOPE ;DISPLAY ORIGIN
        JMP     AGAIN   ;REPEAT PROGRAM


INCRE:  10
LIMIT:  2000
TEMP:   0


        .END    INIT
```

This page intentionally left blank.

APPENDICES

- I/O DEVICE CODES AND DATA GENERAL MNEMONICS
- OCTAL AND HEXADECIMAL CONVERSION
- ASCII - 128 CHARACTER CODES
  EBCDIC CHARACTER CODES
- NOVA LINE COMPUTERS
  INSTRUCTION EXECUTION TIMES

  ECLIPSE COMPUTER
  INSTRUCTION EXECUTION TIMES

This page intentionally left blank

APPENDIX A
I/O DEVICE CODES AND DATA GENERAL MNEMONICS

Device Code (Octal)	Mnemonic	Priority Mask Bit	Device
00	--	--	Power fail
01◑	WCS	--	Writeable control store
02◑	ERCC	--	Error checking and correction
03◑	MAP	--	Memory Allocation and Protection
01○	MDV	--	Multiply/Divide
02○	MMPU	--	Memory Management and Protection Unit
02*○	MAP0	--	Memory Allocation and Protection
03○	MAP1	--	
04○	MAP2	--	
05			
06	MCAT	12	Multiprocessor adapter transmitter
07	MCAR	12	Multiprocessor adapter receiver
10	TTI	14	Teletype input
11	TTO	15	Teletype output
12	PTR	11	Paper tape reader
13	PTP	13	Paper tape punch
14	RTC	13	Real time clock option
15	PLT	12	Incremental plotter
16	CDR	10	Card reader
17	LPT	12	Line printer
20	DSK	9	Fixed head disc
21	ADCV	8	A/D converter
22	MTA	10	Magnetic tape
23	DACV	--	D/A converter
24	DCM	0	Data communications multiplexor
25			
26			
27			
30	QTY	14	Asynchronous hardware multiplexor
31*	IBM1	13	IBM 360/370 interface
32	IBM2		
33	DKP	7	Moving head disc
34	CAS	10	Cassette tape
34*	MX1	11	Multiline asynchronous controller
35	MX2		
36	IPB	6	Interprocessor bus--half-duplex
37	IVT	6	IPB watchdog timer
40	DPI	8	IPB full-duplex input
41	DPO	8	IPB full-duplex output
40+	SCR	8	Synchronous communication receiver
41·	SCT	8	Synchronous communication transmitter

◑ ECLIPSE computer only

○ NOVA line computers only

* code returned by INTA and used by VCT for ECLIPSE computer

+ may be set up with any unused even device code 40 or greater

· may be set up with any unused odd device code 41 or greater

APPENDIX A (Continued) I/O DEVICE CODES AND DATA GENERAL MNEMONICS

Device Code (Octal)	Mnemonic	Priority Mask Bit	Device
42	DIO	7	Digital I/O
43	DIOT	6	Digital I/O timer
44	MXM	12	Modem control for multiline asynchronous controller
45			
46	MCAT1	12	Second multiprocessor transmitter
47	MCAR1	12	Second multiprocessor receiver
50	TTI1	14	Second teletype input
51	TTO1	15	Second teletype output
52	PTR1	11	Second paper tape reader
53	PTP1	13	Second paper tape punch
54	RTC1	13	Second real time clock option
55	PLT1	12	Second incremental plotter
56	CDR1	10	Second card reader
57	LPT1	12	Second line printer
60	DSK1	9	Second fixed head disc
61			
62	MTA1	10	Second magnetic tape
63			
64*^{o}	FPU1	5	Alternate location for floating point
65^{o}	FPU2		
66^{o}	FPU4		
67			
70	QTY1	14	Second asynchronous hardware multiplexor
70	SLA1	14	Second synchronous line adapter
71*		13	Second IBM 360/370 interface
72			
73	DKP1	7	Second moving head disc
74	CAS1	10	Second cassette tape
74*		11	Second multiline asynchronous controller
75			
74*^{o}	FPU1	5	Floating point
75^{o}	FPU2		
76^{o}	FPU		
77	CPU	--	Central processor and console functions

*code returned by INTA and used by VCT for ECLIPSE computer

^{o}NOVA line computers only

APPENDIX B
OCTAL AND HEXADECIMAL CONVERSION

To convert a number from octal or hexadecimal to decimal, locate in each column of the appropriate table the decimal equivalent for the octal or hex digit in that position. Add the decimal equivalents to obtain the decimal number.

1. Locate the largest decimal value in the appropriate table that will fit into the decimal number to be converted;
2. note its octal or hex equivalent and column position;
3. find the decimal remainder.

Repeat the process on each remainder. When the remainder is 0, all digits will have been generated.

	8^5	8^4	8^3	8^2	8^1	8^0
0	0	0	0	0	0	0
1	32,768	4,096	512	64	8	1
2	65,536	8,192	1,024	128	16	2
3	98,304	12,228	1,536	192	24	3
4	131,072	16,384	2,048	256	32	4
5	163,840	20,480	2,560	320	40	5
6	196,608	24,576	3,072	384	48	6
7	229,376	28,672	3,584	448	56	7

	16^5	16^4	16^3	16^2	16^1	16^0
0	0	0	0	0	0	0
1	1,048,576	65,536	4,096	256	16	1
2	2,097,152	131,072	8,192	512	32	2
3	3,145,728	196,608	12,288	768	48	3
4	4,194,304	262,144	16,384	1,024	64	4
5	5,242,880	327,680	20,480	1,280	80	5
6	6,291,456	393,216	24,576	1,536	96	6
7	7,340,032	458,752	28,672	1,792	112	7
8	8,388,608	524,288	32,768	2,048	128	8
9	9,437,184	589,824	36,864	2,304	144	9
A	10,485,760	655,360	40,960	2,560	160	10
B	11,534,336	720,896	45,056	2,816	176	11
C	12,582,912	786,432	49,152	3,072	192	12
D	13,631,488	851,968	53,248	3,328	208	13
E	14,680,064	917,504	57,344	3,584	224	14
F	15,728,640	983,040	61,440	3,840	240	15

This page intentionally left blank.

APPENDIX C
ASCII - 128 CHARACTER CODES

Decimal	Hexa-decimal	7-Bit Octal	ASCII Graphic Or Control Characters	Explanation	To Produce On TTY Mod 33,35	To Produce On TTY Mod 37	ASHCII Punched Card Code	8-Bit Octal
0	00	000	NUL	Null	P°+	Null*	12- 0-1-8-9	000
1	01	001	SOH	Start of Heading	A°	A°	12- 1-9	201
2	02	002	STX	Start of Text	B°	B°	12- 2-9	202
3	03	003	ETX	End of Text	C°	C°	12- 3-9	003
4	04	004	EOT	End of Transmission	D°	D°	7- 9	204
5	05	005	ENQ	Enquiry	E°	E°	0- 5-8-9	005
6	06	006	ACK	Acknowledge	F°	F°	0- 6-8-9	006
7	07	007	BEL	Bell	G°	G°	0- 7-8-9	207
8	08	010	BS	Back Space	H°	backspace*	11- 6-9	210
9	09	011	HT	Horizontal Tab	I°	tab	12- 5-9	011
10	0A	012	LF	Line Feed	line feed			012
					J°	new line*	0- 5-9	012
					line feed°			212
11	0B	013	VT	Vertical Tab	K°	K°	12- 3-8-9	213
12	0C	014	FF	Form Feed	L°	L°	12- 4-8-9	014
13	0D	015	CR	Carriage Return	return	return	12- 5-8-9	215
					M°			215
					return°			015
14	0E	016	SO	Shift Out	N°	N°	12- 6-8-9	216
15	0F	017	SI	Shift In	O°	O°	12- 7-8-9	017
16	10	020	DLE	Data Link Escape	P°	P°	12-11-1-8-9	220
17	11	021	DC1	Device Control 1	Q°	Q°	11- 1-9	021
18	12	022	DC2	Device Control 2	R°	R°	11- 2-9	022
19	13	023	DC3	Device Control 3	S°	S°	11- 3-9	223
20	14	024	DC4	Device Control 4	T°	T°	4- 8-9	024
21	15	025	NAK	Negative Acknowledge	U°	U°	5- 8-9	225
22	16	026	SYN	Synchronous Idle	V°	V°	2- 9	226
23	17	027	ETB	End of Transmission Block	W°	W°	0- 6-9	027
24	18	030	CAN	Cancel	X°	X°	11- 8-9	030
25	19	031	EM	End of Medium	Y°	Y°	11- 1-8-9	231
26	1A	032	SUB	Substitute	Z°	Z°	7- 8-9	232
27	1B	033	ESC	Escape	esc	escape	0- 7-9	033
					K°+			033
28	1C	034	FS	File Separator	L°+	\ °	11- 4-8-9	234
29	1D	035	GS	Group Separator	M°+	]	11- 5-8-9	035
30	1E	036	RS	Record Separator	N°+	^ °	11- 6-8-9	036
31	1F	037	US	Unit Separator	O°+	-°	11- 7-8-9	237
32	20	040	SP	Space	space	space	No Punches	240
33	21	041	!		1+	1+	12- 7-8	041
34	22	042	"		2+	2+	7- 8	042

DG-01016 (Sheet 1)

° Control
+ Shift
* Repeat

APPENDIX C (Continued)
ASCII - 128 CHARACTER CODES

Decimal	Hexa-decimal	7-Bit Octal	ASCII Graphic Or Control Characters	Explanation	To Produce On TTY Mod 33,35	To Produce On TTY Mod 37	ASHCII Punched Card Code	8-Bit Octal
35	23	043	#		3+	3+	3- 8	243
36	24	044	$		4+	4+	11- 3-8	044
37	25	045	%		5+	5+	0- 4-8	245
38	26	046	&		6+	6+	12	246
39	27	047	'		7+	7+	5- 8	047
40	28	050	(		8+	8+	12- 5-8	050
41	29	051	)		9+	9+	11- 5-8	251
42	2A	052	*		: +*	: + *	11- 4-8	252
43	2B	053	+		; +	; +	12- 6-8	053
44	2C	054	,		,	,	0- 3-8	054
45	2D	055	-		-*	-*	11	055
46	2E	056	.		. *	. *	12- 3-8	056
47	2F	057	/		/	/	0- 1	257
48	30	060	0		0	0	0	060
49	31	061	1		1	1	1	261
50	32	062	2		2	2	2	262
51	33	063	3		3	3	3	063
52	34	064	4		4	4	4	264
53	35	065	5		5	5	5	065
54	36	066	6		6	6	6	066
55	37	067	7		7	7	7	267
56	38	070	8		8	8	8	270
57	39	071	9		9	9	9	071
58	3A	072	:		:	:*	2- 8	072
59	3B	073	;		;	;	11- 6-8	273
60	3C	074	<		, +	<	12- 4-8	074
61	3D	075	=		-+	-	6- 8	275
62	3E	076	>		. +	>	0- 6-8	276
63	3F	077	?		/+	/	0- 7-8	077
64	40	100	@		P+	@	4- 8	300
65	41	101	A		A	A+	12- 1	101
66	42	102	B		B	B+	12- 2	102
67	43	103	C		C	C+	12- 3	303
68	44	104	D		D	D+	12- 4	104
69	45	105	E		E	E+	12- 5	305
70	46	106	F		F	F+	12- 6	306
71	47	107	G		G	G+	12- 7	107
72	48	110	H		H	H+	12- 8	110
73	49	111	I		I	I +	12- 9	311
74	4A	112	J		J	J +	11- 1	312

DG-01016 (Sheet 2)

+ Shift
* Repeat

APPENDIX C (Continued)
ASCII - 128 CHARACTER CODES

Decimal	Hexa-decimal	7-Bit Octal	ASCII Graphic Or Control Characters	Explanation	To Produce On TTY Mod 33,35	To Produce On TTY Mod 37	ASHCII Punched Card Code	8-Bit Octal
75	4B	113	K		K	K^{+}	11- 2	113
76	4C	114	L		L	L^{+}	11- 3	314
77	4D	115	M		M	M^{+}	11- 4	115
78	4E	116	N		N	N^{+}	11- 5	116
79	4F	117	O		O	O^{+}	11- 6	317
80	50	120	P		P	P^{+}	11- 7	120
81	51	121	Q		Q	Q^{+}	11- 8	321
82	52	122	R		R	R^{+}	11- 9	322
83	53	123	S		S	S^{+}	0- 2	123
84	54	124	T		T	T^{+}	0- 3	324
85	55	125	U		U	U^{+}	0- 4	125
86	56	126	V		V	V^{+}	0- 5	126
87	57	127	W		W	W^{+}	0- 6	327
88	58	130	X		X	X^{+*}	0- 7	330
89	59	131	Y		Y	Y^{+}	0- 8	131
90	5A	132	Z		Z	Z^{+}	0- 9	132
91	5B	133	[		K^{+}	[	12- 2-8	333
92	5C	134	\		L^{+}	\	0- 2-8	134
93	5D	135	]		M^{+}	]	11- 2-8	335
94	5E	136	↑ (^)		N^{+}	^	11- 7-8	336
95	5F	137	← (-)		O^{+}	-*	0- 5-8	137
96	60	140	`			@$^{+}$	1- 8	140
97	61	141	a			A	12- 0-1	341
98	62	142	b			B	12- 0-2	342
99	63	143	c			C	12- 0-3	143
100	64	144	d			D	12- 0-4	344
101	65	145	e			E	12- 0-5	145
102	66	146	f			F	12- 0-6	146
103	67	147	g			G	12- 0-7	347
104	68	150	h			H	12- 0-8	350
105	69	151	i			I	12- 0-9	151
106	6A	152	j			J	12-11-1	152
107	6B	153	k			K	12-11-2	353
108	6C	154	l			L	12-11-3	154
109	6D	155	m			M	12-11-4	355
110	6E	156	n			N	12-11-5	356
111	6F	157	o			O	12-11-6	157
112	70	160	p			P	12-11-7	360
113	71	161	q			Q	12-11-8	161
114	72	162	r			R	12-11-9	162

DG-01016 (Sheet 3)

+ Shift
* Repeat

APPENDIX C (Continued)
ASCII - 128 CHARACTER CODES

Decimal	Hexa-decimal	7-Bit Octal	ASCII Graphic Or Control Characters	Explanation	To Produce On TTY Mod 33,35	To Produce On TTY Mod 37	ASHCII Punched Card Code	8-Bit Octal
115	73	163	s			S	11- 0-2	363
116	74	164	t			T	11- 0-3	164
117	75	165	u			U	11- 0-4	365
118	76	166	v			V	11- 0-5	366
119	77	167	w			W	11- 0-6	167
120	78	170	x			X*	11- 0-7	170
121	79	171	y			Y	11- 0-8	371
122	7A	172	z			Z	11- 0-9	372
123	7B	173	{			[+	12- 0	173
124	7C	174	|			\+	12-11	374
125	7D	175	}			]	11- 0	175
126	7E	176	~			^+	11- 0-1	176
127	7F	177	DEL	Delete	rubout	delete*	12- 7-9	377

DG-01016 (Sheet 4)

+ Shift
* Repeat

APPENDIX C (Continued)
EBCDIC CHARACTER CODES

Decimal	Hexadecimal	EBCDIC Graphic or Control Characters	Explanation	Punched Card Code
0	00	NUL	Null	12- 0-1-8-9
1	01	SOH	Start of Heading	12- 1-9
2	02	STX	Start of Text	12- 2-9
3	03	EXT	End of Text	12- 3-9
4	04	PF	Punch Off	12- 4-9
5	05	HT	Horizontal Tab	12- 5-9
6	06	LC	Lower Case	12- 6-9
7	07	DEL	Delete	12- 7-9
8	08			12- 8-9
9	09			12- 1-8-9
10	0A	SMM	Start of Manual Message	12- 2-8-9
11	0B	VT	Vertical Tab	12- 3-8-9
12	0C	FF	Form Feed	12- 4-8-9
13	0D	CR	Carriage Return	12- 5-8-9
14	0E	SO	Shift Out	12- 6-8-9
15	0F	SI	Shift In	12- 7-8-9
16	10	DLE	Data Link Escape	12-11-1-8-9
17	11	DC1	Device Control 1	11- 1-9
18	12	DC2	Device Control 2	11- 2-9
19	13	TM	Tape Mark	11- 3-9
20	14	RES	Restore	11- 4-9
21	15	NL	New Line	11- 5-9
22	16	BS	Back Space	11- 6-9
23	17	IL	Idle	11- 7-9
24	18	CAN	Cancel	11- 8-9
25	19	EM	End of Medium	11- 1-8-9
26	1A	CC	Cursor Control	11- 2-8-9
27	1B	CU1	Customer Use 1	11- 3-8-9
28	1C	IFS	Interchange File Separator	11- 4-8-9
29	1D	IGS	Interchange Group Separator	11- 5-8-9
30	1E	IRS	Interchange Record Separator	11- 6-8-9
31	1F	IUS	Interchange Unit Separator	11- 7-8-9
32	20	DS	Digit Select	11- 0-1-8-9
33	21	SOS	Start of Significance	0- 1-9
34	22	FS	Field Separator	0- 2-9
35	23			0- 3-9
36	24	BYP	Bypass	0- 4-9
37	25	LF	Line Feed	0- 5-9
38	26	ETB	End of Transmission Block	0- 6-9
39	27	ESC	Escape	0- 7-9
40	28			0- 8-9
41	29			0- 1-8-9
42	2A	SM	Set Mode	0- 2-8-9
43	2B	CU2	Customer Use 2	0- 3-8-9
44	2C			0- 4-8-9

APPENDIX C (Continued)
EBCDIC CHARACTER CODES

Decimal	Hexadecimal	EBCDIC Graphic or Control Characters	Explanation	Punched Card Code
45	2D	ENQ	Enquiry	0- 5-8-9
46	2E	ACK	Acknowledge	0- 6-8-9
47	2F	BEL	Bell	0- 7-8-9
48	30			12-11-0-1-8-9
49	31			1- 9
50	32	SYN	Synchronous Idle	2- 9
51	33			3- 9
52	34	PN	Punch On	4- 9
53	35	RS	Reader Stop	5- 9
54	36	UC	Upper Case	6- 9
55	37	EOT	End of Transmission	7- 9
56	38			8- 9
57	39			1- 8-9
58	3A			2- 8-9
59	3B	CU3	Customer Use 3	3- 8-9
60	3C	DC4	Device Control 4	4- 8-9
61	3D	NAK	Negative Acknowledge	5- 8-9
62	3E			6- 8-9
63	3F	SUB	Substitute	7- 8-9
64	40	SP	Space	no punches
65	41			12- 0-1-9
66	42			12- 0-2-9
67	43			12- 0-3-9
68	44			12- 0-4-9
69	45			12- 0-5-9
70	46			12- 0-6-9
71	47			12- 0-7-9
72	48			12- 0-8-9
73	49			12- 1-8
74	4A	¢		12- 2-8
75	4B	.		12- 3-8
76	4C	<		12- 4-8
77	4D	(		12- 5-8
78	4E	+		12- 6-8
79	4F	|		12- 7-8
80	50	&		12
81	51			12-11-1-9
82	52			12-11-2-9
83	53			12-11-3-9
84	54			12-11-4-9
85	55			12-11-5-9
86	56			12-11-6-9
87	57			12-11-7-9
88	58			12-11-8-9

APPENDIX C (Continued)
EBCDIC CHARACTER CODES

Decimal	Hexadecimal	EBCDIC Graphic or Control Characters	Explanation	Punched Card Code
89	59			11- 1-8
90	5A	!		11- 2-8
91	5B	$		11- 3-8
92	5C	*		11- 4-8
93	5D	)		11- 5-8
94	5E			11- 6-8
95	5F	¬		11- 7-8
96	60	-		11
97	61	/		0- 1
98	62			11- 0-2-9
99	63			11- 0-3-9
100	64			11- 0-4-9
101	65			11- 0-5-9
102	66			11- 0-6-9
103	67			11- 0-7-9
104	68			11- 0-8-9
105	69			0- 1-8
106	6A	¦		12-11
107	6B	.		0- 3-8
108	6C	%		0- 4-8
109	6D	–		0- 5-8
110	6E	>		0- 6-8
111	6F	?		0- 7-8
112	70			12-11-0
113	71			12-11-0-1-9
114	72			12-11-0-2-9
115	73			12-11-0-3-9
116	74			12-11-0-4-9
117	75			12-11-0-5-9
118	76			12-11-0-6-9
119	77			12-11-0-7-9
120	78			12-11-0-8-9
121	79			1- 8
122	7A	:		2- 8
123	7B	#		3- 8
124	7C	@		4- 8
125	7D	.		5- 8
126	7E	=		6- 8
127	7F	"		7- 8
128	80			12- 0-1-8
129	81	a		12- 0-1
130	82	b		12- 0-2
131	83	c		12- 0-3
132	84	d		12- 0-4

APPENDIX C (Continued)
EBCDIC CHARACTER CODES

Decimal	Hexadecimal	EBCDIC Graphic or Control Characters	Explanation	Punched Card Code
133	85	e		12- 0-5
134	86	f		12- 0-6
135	87	g		12- 0-7
136	88	h		12- 0-8
137	89	i		12- 0-9
138	8A			12- 0-2-8
139	8B			12- 0-3-8
140	8C			12- 0-4-8
141	8D			12- 0-5-8
142	8E			12- 0-6-8
143	8F			12- 0-7-8
144	90			12-11-1-8
145	91	j		12-11-1
146	92	k		12-11-2
147	93	l		12-11-3
148	94	m		12-11-4
149	95	n		12-11-5
150	96	o		12-11-6
151	97	p		12-11-7
152	98	q		12-11-8
153	99	r		12-11-9
154	9A			12-11-2-8
155	9B			12-11-3-8
156	9C			12-11-4-8
157	9D			12-11-5-8
158	9E			12-11-6-8
159	9F			12-11-7-8
160	A0			11- 0-1-8
161	A1	~		11- 0-1
162	A2	s		11- 0-2
163	A3	t		11- 0-3
164	A4	u		11- 0-4
165	A5	v		11- 0-5
166	A6	w		11- 0-6
167	A7	x		11- 0-7
168	A8	y		11- 0-8
169	A9	z		11- 0-9
170	AA			11- 0-2-8
171	AB			11- 0-3-8
172	AC			11- 0-4-8
173	AD			11- 0-5-8
174	AE			11- 0-6-8
175	AF			11- 0-7-8
176	B0			12-11-0-1-8
177	B1			12-11-0-1

APPENDIX C (Continued)
EBCDIC CHARACTER CODES

Decimal	Hexadecimal	EBCDIC Graphic or Control Characters	Explanation	Punched Card Code
178	B2			12-11-0-2
179	B3			12-11-0-3
180	B4			12-11-0-4
181	B5			12-11-0-5
182	B6			12-11-0-6
183	B7			12-11-0-7
184	B8			12-11-0-8
185	B9			12-11-0-9
186	BA			12-11-0-2-8
187	BB			12-11-0-3-8
188	BC			12-11-0-4-8
189	BD			12-11-0-5-8
190	BE			12-11-0-6-8
191	BF			12-11-0-7-8
192	C0	{		12- 0
193	C1	A		12- 1
194	C2	B		12- 2
195	C3	C		12- 3
196	C4	D		12- 4
197	C5	E		12- 5
198	C6	F		12- 6
199	C7	G		12- 7
200	C8	H		12- 8
201	C9	I		12- 9
202	CA			12- 0-2-8-9
203	CB			12- 0-3-8-9
204	CC	⑀		12- 0-4-8-9
205	CD			12- 0-5-8-9
206	CE	⑂		12- 0-6-8-9
207	CF			12- 0-7-8-9
208	D0	}		11- 0
209	D1	J		11- 1
210	D2	K		11- 2
211	D3	L		11- 3
212	D4	M		11- 4
213	D5	N		11- 5
214	D6	O		11- 6
215	D7	P		11- 7
216	D8	Q		11- 8
217	D9	R		11- 9
218	DA			12-11-2-8-9
219	DB			12-11-3-8-9
220	DC			12-11-4-8-9
221	DD			12-11-5-8-9

APPENDIX C (Continued)
EBCDIC CHARACTER CODES

Decimal	Hexadecimal	EBCDIC Graphic or Control Characters	Explanation	Punched Card Code
222	DE			12-11-6-8-9
223	DF			12-11-7-8-9
224	E0			0- 2-8
225	E1			11- 0-1-9
226	E2	S		0- 2
227	E3	T		0- 3
228	E4	U		0- 4
229	E5	V		0- 5
230	E6	W		0- 6
231	E7	X		0- 7
232	E8	Y		0- 8
233	E9	Z		0- 9
234	EA			11- 0-2-8-9
235	EB			11- 0-3-8-9
236	EC	⑁		11- 0-4-8-9
237	ED			11- 0-5-8-9
238	EE			11- 0-6-8-9
239	EF			11- 0-7-8-9
240	F0	0		0
241	F1	1		1
242	F2	2		2
243	F3	3		3
244	F4	4		4
245	F5	5		5
246	F6	6		6
247	F7	7		7
248	F8	8		8
249	F9	9		9
250	FA	LVM		12-11-0-2-8-9
251	FB			12-11-0-3-8-9
252	FC			12-11-0-4-8-9
253	FD			12-11-0-5-8-9
254	FE			12-11-0-6-8-9
255	FF	EO		12-11-0-7-8-9

APPENDIX D

NOVA LINE COMPUTERS INSTRUCTION EXECUTION TIMES

SUPERNOVA read-only time equals semiconductor time, except add 0.2 for LDA, STA, ISZ, and DSZ if reference is to core. NOVA times are for core; for read-only subtract 0.2 except subtract 0.4 for LDA, STA, ISZ, and DSZ if reference is to read-only memory. When two numbers are given, the one at the left of the slash is the time for an isolated transfer, the one at the right is the minimum time between consecutive transfers. All times are in microseconds.

		SUPERNOVA		1200	800,820		NOVA 2	
	NOVA	SC	CORE	SERIES	840	830	8K	16K
LDA	5.2	1.2	1.6	2.55	1.6	2.0	1.6	2.0
STA	5.5	1.2	1.6	2.55	1.6	2.0	1.6	2.0
ISZ, DSZ	5.2	1.4	1.8	3.15	1.8	2.2	1.7	2.1
JMP	2.6	0.6	0.8	1.35	0.8	1.0	0.8	1.0
JSR	3.5	1.2	1.4	1.35	0.8	1.0	1.1	1.2
COM, NEG, MOV, INC	5.6	0.3	0.8	1.35	0.8	1.0	0.8	1.0
ADC, SUB, ADD, AND	5.9	0.3	0.8	1.35	0.8	1.0	0.8	1.0
Each level of @, add	2.6	0.6	0.8	1.2	0.8	1.0	0.8	1.0
Each autoindex, add	0.0	0.2	0.2	0.6	0.2	0.2	0.5	0.5
Base register addr. add	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
If skip occurs, add	0.0	*	0.8	1.35	0.2	0.2	0.3	0.2
I/O input (except INTA)	4.4	2.8	2.9	2.55	2.2	2.4	1.4	1.5
INTA	4.4	3.6	3.7	2.55	2.2	2.4	1.4	1.5
I/O output	4.7	3.2	3.3	3.15	2.2	2.4	1.6	1.7
NIO	4.4	3.2	3.3	3.15	2.2	2.4	1.6	1.7
I/O skips	4.4	2.8	2.9	2.55	1.4	1.6	1.1	1.2
If skip occurs, add	0.0	0.0	0.0	0.0	0.2	0.2	0.3	0.2
For S, C, or P; add	0.0	0.0	0.0	0.0	0.6	0.6	0.3	0.3
MUL								
Average	11.1	3.7	3.8	3.75	8.8	9.0	6.1	6.2
Maximum	11.1	5.3	5.4	3.75	8.8	9.0	6.1	6.2
DIV								
Successful	11.9	6.8	6.9	4.05	8.8	9.0	6.4	6.5
Unsuccessful	11.9	1.5	1.6	2.55	1.6	2.0	6.4	6.5
INTERRUPT LATENCY								
With MUL/DIV	12.0	9.0	9.0	7.0	10.6	12.0	5.8	5.9
Without MUL/DIV	12.0	5.0	5.0	7.0	4.6	6.0	1.9	2.3
DATA CHANNEL								
Input	3.5	2.3	2.3	1.2	2.0	2.2	2.0	2.1
Output	4.4	2.8	2.8	1.2/1.8	2.0	2.2	2.1	2.2
Increment	4.4	2.8	2.8	1.8/2.4	2.2	2.4	2.2	2.3
Add to memory	5.3	2.8	2.8	---	---	N/A	---	---
Latency+								
With MUL/DIV	17.3	11.8	11.8	9.4	5.8	6.4	5.2	5.3
Without MUL/DIV	17.3	7.8	7.8	9.4	5.8	6.4	5.2	5.3
HIGH SPEED DATA CHANNEL								
Input	N/A	0.8	0.8	N/A	0.8	1.0	0.8	0.9/1.0
Output		0.8/1.0	0.8/1.0		0.8/1.0	1.0/1.2	1.2	1.3
Increment		1.0/1.2	1.0/1.2		1.0/1.2	1.2/1.4	1.3	1.4
Add to memory		1.0/1.2	1.0/1.2		---	N/A	---	---
Latency+								
With MUL/DIV		5.7	5.7		4.8	5.4	4.3	4.4
Without MUL/DIV		3.7	3.7		3.2	3.6	4.3	4.4

*If 2AC-multiple operation instruction is skipped, add 0.3; otherwise add 0.6.

+For highest priority peripheral on I/O bus.

APPENDIX D (Continued)

ECLIPSE COMPUTER INSTRUCTION EXECUTION TIMES

The following table gives minimum, maximum, and typical execution times for all instructions in the basic instruction set. These times assume a system without the MAP feature operating with 4-way interleaved core memory. All times are in microseconds.

STANDARD INSTRUCTION SET	MINIMUM	MAXIMUM	TYPICAL	NOTES
ADD	0.6	0.6	0.6	1
ADD COMPLEMENT	0.6	0.6	0.6	1
ADD IMMEDIATE	0.6	0.6	0.6	
EXTENDED ADD IMMEDIATE	1.2	1.2	1.2	
AND	0.6	0.6	0.6	1
AND IMMEDIATE	1.2	1.2	1.2	
AND WITH COMPLEMENTED SOURCE	0.6	0.6	0.6	
BLOCK ADD AND MOVE	1.8+0.8N	1.8+1.2N	1.7 +1.0N	2
BLOCK MOVE	2.0+0.6N	1.4+1.2N	1.85+0.85N	2
COMPARE LIMITS	specified AC's not the same			
number within limits	1.8	2.2	1.9	
number less than L	2.0	2.4	2.1	
number greater than H	2.2	2.6	2.3	
	specified AC's the same			
number within limits	1.6	1.6	1.6	
number less than L	1.8	1.8	1.8	
number greater than H	1.6	1.6	1.6	
COMPLEMENT	0.6	0.6	0.6	1
COUNT BITS	1.0	10.6	1.0 +0.6N	3
DECIMAL ADD	0.6	0.6	0.6	
DECREMENT AND SKIP IF ZERO	1.4	1.6	1.5	4
EXTENDED DECREMENT AND SKIP IF ZERO	2.4	2.6	2.45	4
DECIMAL SUBTRACT	0.6	0.6	0.6	
DISPATCH				
number within limits	3.6	3.6	3.6	4, 11
number less than L	2.8	2.8	2.8	4
number greater than H	2.6	2.8	2.65	
DOUBLE HEX SHIFT LEFT	2.4	4.2	depends on shift count	
DOUBLE HEX SHIFT RIGHT	2.4	4.2	depends on shift count	
DOUBLE LOGICAL SHIFT	1.0	5.6	depends on shift count	
ENTER WCS	depends on user instruction			
EXCHANGE ACCUMULATORS	0.8	0.8	0.8	
EXCLUSIVE OR	0.6	0.6	0.6	
EXCLUSIVE OR IMMEDIATE	1.2	1.2	1.2	
EXECUTE	0.8 + time for instruction to be executed			
EXTENDED OPERATION	4.8	5.8	5.05	5
HALVE	1.0	1.0	1.0	
HEX SHIFT LEFT	1.8	3.0	depends on shift count	
HEX SHIFT RIGHT	1.8	3.0	depends on shift count	
INCLUSIVE OR	0.6	0.6	0.6	
INCLUSIVE OR IMMEDIATE	1.2	1.2	1.2	
INCREMENT	0.6	0.6	0.6	1
INCREMENT AND SKIP IF ZERO	1.4	1.6	1.5	4
EXTENDED INCREMENT AND SKIP IF ZERO	2.4	2.6	2.45	4

APPENDIX D (Continued)
ECLIPSE COMPUTER
INSTRUCTION EXECUTION TIMES

STANDARD INSTRUCTION SET	MINIMUM	MAXIMUM	TYPICAL	NOTES
JUMP	0.6	0.8	0.65	4
EXTENDED JUMP	1.6	1.6	1.6	4
JUMP TO SUBROUTINE	0.6	0.8	0.65	4
EXTENDED JUMP TO SUBROUTINE	1.6	1.6	1.6	4
LOAD ACCUMULATOR	0.8	1.4	1.0	4
EXTENDED LOAD ACCUMULATOR	1.8	2.4	1.95	4
EXTENDED LOAD EFFECTIVE ADDRESS	1.6	1.6	1.6	4
LOAD BYTE	1.4	1.8	1.5	
LOAD MAP	2.4+0.6N	2.4+0.6N	2.3+0.6N	2
LOCATE AND RESET LEAD BIT	1.2	7.2	1.2+0.4N	3
LOCATE LEAD BIT	1.0	7.0	1.0+0.4N	3
LOGICAL SHIFT	1.0	3.8	depends on shift count	
MODIFY STACK POINTER	2.2	2.4	2.25	12
MOVE	0.6	0.6	0.6	1
NEGATE	0.6	0.6	0.6	1
POP BLOCK	4.0	4.8	4.2	7
POP MULTIPLE ACCUMULATORS	2.2+0.4N	3.0+0.4N	2.4+0.4N	7, 8
POP PC AND JUMP	2.4	3.2	2.6	7
PUSH JUMP	2.2	3.6	2.7	6, 13
PUSH MULTIPLE ACCUMULATORS	2.2+0.4N	3.0+0.4N	2.4+0.4N	5, 8
PUSH RETURN ADDRESS	2.6	3.2	2.8	5
RESTORE	6.0	8.0	6.85	
RETURN	4.4	5.0	4.55	7
SAVE	3.8	5.2	4.08	5
SET BIT TO ONE	2.4	2.8	2.45	6
SET BIT TO ZERO	2.4	2.8	2.45	6
SIGN EXTEND AND DIVIDE	2.2	9.8	9.5	
SIGNED DIVIDE	2.2	10.2	9.6	
SIGNED MULTIPLY	7.2	7.2	7.2	
SKIP IF ACS > ACD	1.0	1.0	1.0	
SKIP IF ACS ≥ ACD	1.0	1.0	1.0	
SKIP ON NON-ZERO BIT	2.2	2.6	2.3	6, 10
SKIP ON ZERO BIT	2.2	2.6	2.3	6, 10
SKIP ON ZERO BIT AND SET TO ONE	2.6	2.8	2.8	6
STORE ACCUMULATOR	0.8	1.4	1.0	4
EXTENDED STORE ACCUMULATOR	1.8	2.4	1.95	4
STORE BYTE	1.8	2.0	1.85	
SUBTRACT	0.6	0.6	0.6	1
SUBTRACT IMMEDIATE	0.6	0.6	0.6	
SYSTEM CALL	4.2	5.0	4.45	5, 6
UNSIGNED DIVIDE	1.6	8.2	8.2	
UNSIGNED MULTIPLY	7.2	7.2	7.2	

APPENDIX D (Continued)

ECLIPSE COMPUTER INSTRUCTION EXECUTION TIMES

I/O INSTRUCTION SET	MINIMUM	MAXIMUM	TYPICAL	NOTES
DATA INPUT	2.2	2.2	2.2	9
DATA OUTPUT	2.6	2.6	2.6	9
I/O SKIP	0.8	0.8	0.8	10
NO I/O TRANSFER	1.2	1.2	1.2	9
VECTOR ON INTERRUPTING DEVICE CODE				
MODE A	2.6	2.8	2.65	
MODE B	8.6	9.6	8.85	5
MODE C	10.2	12.2	10.75	5
MODE D	15.0	18.0	16.5	5
MODE E	16.6	20.2	18.05	5
PROGRAM INTERRUPT CYCLE	1.4	1.8	1.5	4
DATA CHANNEL INPUT	0.8	0.8	0.8	
DATA CHANNEL OUTPUT	1.4	1.6	1.6	
DATA CHANNEL LATENCY+	4.1	4.3	4.3	

PROGRAM INTERRUPT LATENCY is the sum of the longest time that the program runs with the interrupt system disabled plus the time for the program interrupt cycle. The longest non-interruptable instruction is a Mode E VECTOR with a maximum time of 20.2 microseconds. Because this VECTOR also enables the interrupt system, an interrupt will not be honored until after the next instruction, unless the next instruction is interruptable (e.g., BLOCK ADD AND MOVE). Therefore, the time for the next longest instruction must be added. The next longest instruction is COUNT BITS WITH A worst case time of 10.6 microseconds. To this must be added the time for the program interrupt cycle of 1.8 microseconds. This yields an absolute worst case program interrupt latency of 32.6 microseconds. The interrupt latency for a specific application can be computed using the above method.

+For highest priority peripheral on I/O bus.

APPENDIX D (Continued)

ECLIPSE COMPUTER INSTRUCTION EXECUTION TIMES

	MINIMUM	MAXIMUM	TYPICAL
NOTES: 1. If skip occurs, add:	0.6	0.6	0.6
2. N is number of words moved.			
For each indirect reference in AC3, add:	0.8	0.8	0.8
For each indirect reference in AC2, add:	0.6	0.8	0.65
If N is less than 1, then time is:	1.2	1.2	1.2
3. N is the count added to ACD.			
For LOCATE AND RESET LEAD BIT, if the count is 16, the time is:	7.4	7.4	7.4
For LOCATE LEAD BIT, if the count is 16, the time is:	7.2	7.2	7.2
4. For each indirect reference, add:	0.6	0.8	0.65
For each indirect auto-index reference, add:	1.0	1.6	1.15
5. If stack overflows, add:	3.2	3.8	3.45
In addition, see note 6.			
6. For each indirect reference, add:	0.8	0.8	0.8
7. If stack underflows and underflow protection is disabled, add:	0.4	0.8	0.7
If stack underflows and underflow protection is enabled, add:	3.8	5.0	4.4
In addition, see note 6.			
8. N is number of words pushed or popped.			
9. S, C, and P functions require no extra time.			
10. If skip occurs, add:	0.4	0.4	0.4
11. For each indirect reference in the table address add:	0.6	0.6	0.6
12. If stack overflows add:	4.2	4.8	4.65
13. If stack overflows add:	4.0	4.6	4.25

APPENDIX D (Continued)

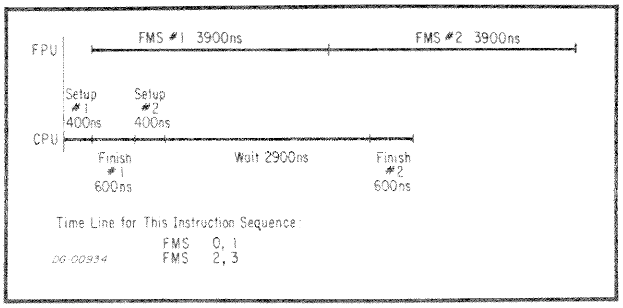
ECLIPSE COMPUTER INSTRUCTION EXECUTION TIMES

FLOATING POINT INSTRUCTION EXECUTION TIMES

Because the CPU and the floating point feature operate in parallel, there are two distinct times to consider when dealing with the execution time of a floating point instruction. These are "FPU time" and "CPU time".

FPU time is the amount of time taken in the floating point unit actually performing the calculation.

CPU time is that amount of time that the CPU devotes to a floating point instruction. This time is divided into three parts: setup time, wait time, and finish time. Setup time is the time devoted to decoding the instruction and computing the effective address if required. Wait time is the time spent by the CPU waiting for the FPU to finish a previous operation and become idle. Finish time is the time devoted to transferring to the FPU all required operands and initiating the floating point operation. The following example illustrates these times.



Wait time is given by the following equation:

WAIT= FPU time for previous instruction - (finish time for previous instruction + total execution time for non-floating point instructions between the floating point instructions + setup time for this floating point instruction).

If WAIT is less than 0, then a value of 0 should be used for WAIT.

APPENDIX D (Continued)

ECLIPSE COMPUTER INSTRUCTION EXECUTION TIMES

INSTRUCTION	CPU		FPU	REMARKS
	SETUP	FINISH		
ADD SINGLE (FPAC)	0.4	0.6	1.5	Exponent over- or underflow
ADD DOUBLE (FPAC)			2.3	Mantissa overflow
SUBTRACT SINGLE (FPAC)			2.4	Normalization needed
SUBTRACT DOUBLE (FPAC)			1.9	Normalization not needed
ADD SINGLE (MEMORY)	1.2	0.8	2.2	Exponent over- or underflow
SUBTRACT SINGLE (MEMORY)	(Note 1)		3.0	Mantissa overflow
			3.1	Normalization needed
			2.6	Normalization not needed
ADD DOUBLE (MEMORY)	1.2	1.4	2.8	Exponent over- or underflow
SUBTRACT DOUBLE (MEMORY)	(Note 1)		3.6	Mantissa overflow
			3.7	Normalization needed
			3.2	Normalization not needed
MULTIPLY SINGLE (FPAC)	0.4	0.6	3.9	
MULTIPLY DOUBLE (FPAC)	0.4	0.6	7.1	
MULTIPLY SINGLE (MEMORY)	1.2	0.8	4.6	Note 1
MULTIPLY DOUBLE (MEMORY)	1.2	1.4	8.4	Note 1
DIVIDE SINGLE (FPAC)	0.4	0.6	4.2	Divisor mantissa > dividend mantissa
			5.1	Divisor mantissa ≤ dividend mantissa
DIVIDE DOUBLE (FPAC)	0.4	0.6	7.4	Divisor mantissa > dividend mantissa
			8.3	Divisor mantissa ≤ dividend mantissa
DIVIDE SINGLE (MEMORY)	1.2	0.8	4.9	Divisor mantissa > dividend mantissa
	(Note 1)		5.8	Divisor mantissa ≤ dividend mantissa
DIVIDE DOUBLE (MEMORY)	1.2	1.4	8.7	Divisor mantissa > dividend mantissa
	(Note 1)		9.6	Divisor mantissa ≤ dividend mantissa
LOAD SINGLE	1.2	0.8	1.6	Note 1
LOAD DOUBLE	1.2	1.4	2.2	Note 1
STORE SINGLE	1.2	0.8	0.5	Notes 1, 2
STORE DOUBLE	1.2	1.2	0.5	Notes 1, 2
FLOAT FROM AC	0.4	0.6	1.9	Integer positive
			2.3	Integer negative
FLOAT FROM MEMORY	1.2	0.8	2.3	Integer positive
	(Note 1)		2.7	Integer negative
FIX TO AC	0.4	0.6	2.1	Integer positive
	(Note 2)		2.5	Integer negative
FIX TO MEMORY	1.2	0.8	2.3	Integer positive
	(Notes 1, 2)		2.7	Integer negative

APPENDIX D (Continued)
ECLIPSE COMPUTER
INSTRUCTION EXECUTION TIMES

	CPU			
INSTRUCTION	SETUP	FINISH	FPU	REMARKS
NEGATE	0.4	0.6	1.3	
ABSOLUTE VALUE	0.4	0.6	1.3	
READ HIGH WORD	0.4	0.6	0.4	Note 2
SCALE	0.6	0.6	1.7	
LOAD EXPONENT	0.6	0.6	1.6	
HALVE	0.8	0.6	1.8	
MOVE	0.4	0.6	1.0	
NORMALIZE	0.4	0.6	1.4	
COMPARE	0.4	0.6	0.9	
LOAD STATUS	1.6	0.8	0.7	Notes 1, 2
STORE STATUS	1.6	0.8	0.5	Notes 1, 2
PUSH FLOATING POINT STATE	1.4	7.0	7.0	
POP FLOATING POINT STATE	1.4	8.4	8.4	
TRAP ENABLE	1.0	0.6	0.4	
TRAP DISABLE				
CLEAR ERRORS				
SKIP TESTS	0.4	0.6	0.4	Note 3

NOTES:

1. For setup time, add 0.4 for first indirect reference and 0.6 for each subsequent indirect reference. For finish time, add 0.2 for each indirect reference except for store instructions, add 0.0 for each indirect reference.
2. FPU time can begin concurrently with the beginning of setup time, if the FPU is idle. Otherwise, FPU time begins as soon as the FPU finishes the previous instruction. Finish time cannot commence until the FPU has completed this instruction.
3. If skip occurs, add 0.2 to finish time.

READERS COMMENT FORM

DOCUMENT TITLE: ....

Your comments, accompanied by answers to the following questions, help us improve the quality and usefulness of our publications. If your answer to a question is "no" or requires qualification, please explain.

How did you use this publication?

( ) As an introduction to the subject.
( ) As an aid for advanced knowledge.
( ) For information about operating procedures.
( ) To instruct in a class.
( ) As a student in a class.
( ) As a reference manual.
( ) Other...

Did you find the material:

- Useful.............................YES ( ) NO ( )
- Complete.........................YES ( ) NO ( )
- Accurate.........................YES ( ) NO ( )
- Well organized..............YES ( ) NO ( )
- Well written...................YES ( ) NO ( )
- Well illustrated..............YES ( ) NO ( )
- Well indexed..................YES ( ) NO ( )
- Easy to read..................YES ( ) NO ( )
- Easy to understand.....YES ( ) NO ( )

We would appreciate any other comments; please label each comment as an addition, deletion, change, or error and reference page numbers where applicable.

COMMENTS

PAGE	COL	PARA	LINE	FROM	TO

From:

NAME.. TITLE..................................
FIRM... DIV. ....
ADDRESS...
CITY... STATE............... ZIP..........
TELEPHONE............................. DATE...................................

Data General Corporation

ENGINEERING PUBLICATIONS
COMMENT FORM
DG-00935

CUT ALONG DOTTED LINE

FOLD DOWN FIRST FOLD DOWN

FOLD UP SECOND FOLD UP

STAPLE