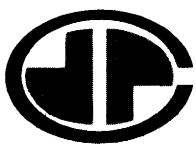
CDP-XI/00 PROCESSOR
MAINTENANCE MANUAL

VOLUME I



california data processors

2019 south ritchey street · santa ana, california 92705 · (714) 558-8211

CDP-XI/00 PROCESSOR
MAINTENANCE MANUAL

VOLUME I

DOCUMENT 21518003
Revision X2
May 1974

california data processors

$14.00

CONTENTS

Page

SECTION 5: (Continued)

SECTION 6: INSTALLATION

SECTION 7: MAINTENANCE

APPENDICES

TABLES

ILLUSTRATIONS

SECTION 1
INTRODUCTION

1.1 PURPOSE AND SCOPE

This manual provides the information needed to understand, install and maintain the Cal Data CDP-XI/00 Processor when used with Drawing Package 21518004. The information in this manual is for the use of a skilled technician familiar with standard test equipment, solid-state logic theory, common maintenance practices and standard troubleshooting techniques.

A basic knowledge of design principles and circuits used in small computers is assumed, hence no tutorial material of this kind is included.

As a stand-alone publication, this manual has a good functional and physical description of the CDP-XI/00 processor, providing the information needed to understand the capabilities and optional features of the CDP-XI and to plan a system using it. The maintenance coverage of this manual is commensurate with the prerequisite skills and knowledge of the defined user, characteristics of the product and maintainability requirements established by Cal Data.

Users holding controlled copies will be provided with revisions and additions to this manual.

1.2 DOCUMENTATION

Major functional areas covered in this manual are divided into two volumes. Volume I covers:

a. CDP-XI/00 Processor.
b. MACROBUS TM.
c. Macropanel.

Volume II covers (as required):

a. Emulation package for a custom CDP-XI model.
b. CDP-XI/MMU Memory Management Unit.
c. CDP-XI processor options.

1.2.1 Publications

Figure 1-1 illustrates the relationship between CDP-XI system hardware and technical documentation. Controlled copies of documents are delivered in accordance with the terms of the purchase contract and are kept current for the life of the product.

The following CDP-XI system documentation is provided, as applicable, for a particular user configuration.

TM - MACROBUS is a trademark of California Data Processors.

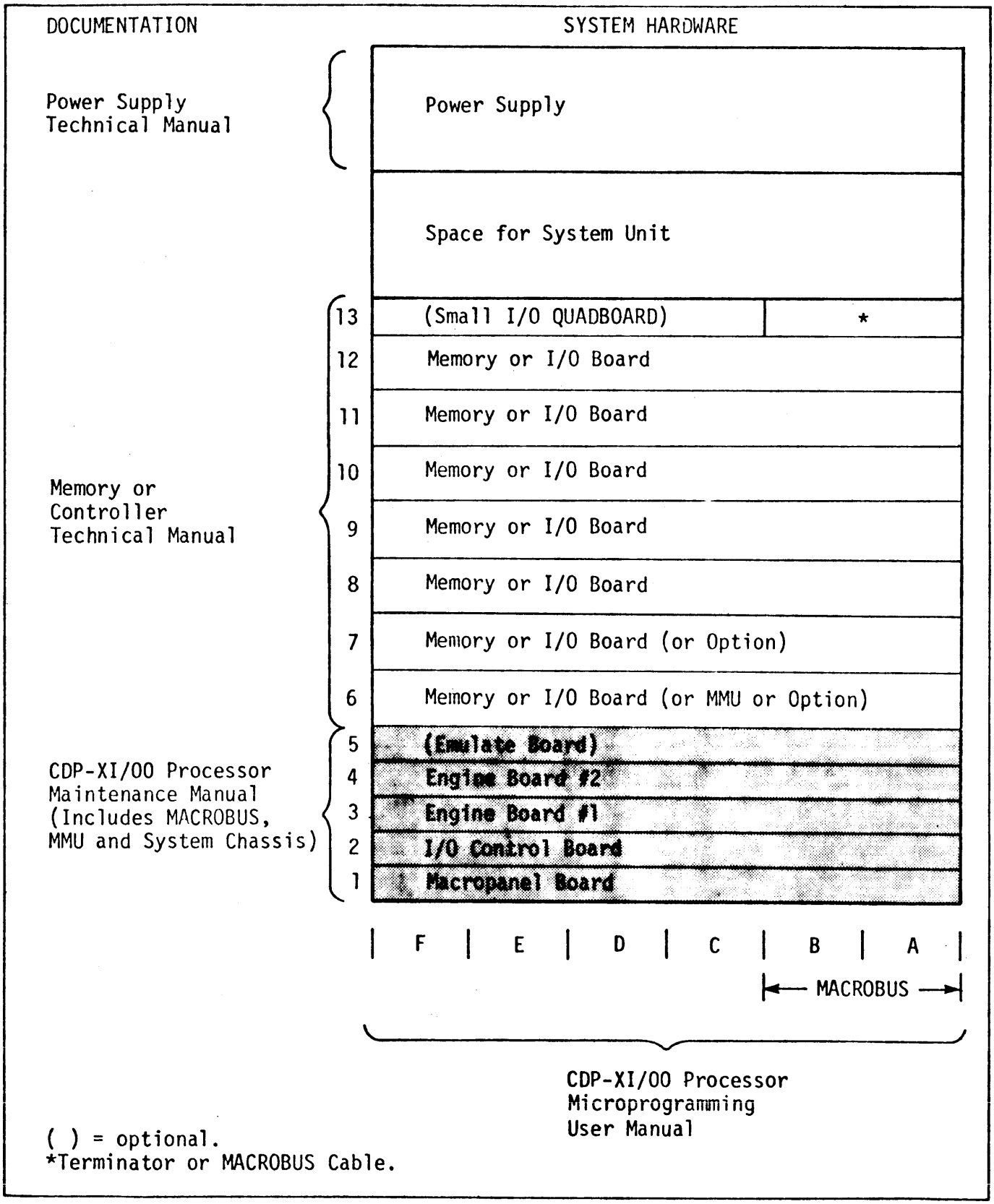


Figure 1-1. Relationship of CDP-XI System Documentation to Hardware

1.2.1.1 CDP-XI/00 Processor Maintenance Manual

This manual provides technical design detail, installation and maintenance information on the system. The MACROBUS, macropanel and chassis are covered in this manual as well as processor options, such as MMU. The technical manual is supported by a controlled drawing package that includes:

a. Schematics
b. Board assembly drawings
c. Theory of operation
d. Firmware listing
e. Mechanical drawings
f. Parts list

1.2.1.2 CDP-XI/00 Processor Microprogramming User Manual

This manual provides the information needed for a programmer to microprogram the CDP-XI/00 processor. Topics covered include:

a. Functional description of the processor and MACROBUS.
b. Microcommand structure and repertoire.
c. Use of the macropanel.
d. Use of the symbolic microassembler and assembly language.

1.2.1.3 CDP-XI Processor Emulation User Manual

This manual describes the macrolevel capabilities of a particular CDP-XI model, and serves as the user's reference manual for programming and operation of the system.

1.2.1.4 Magnetic Core Memory Technical Manuals

These manuals provide design detail, installation and maintenance information on the CDP-8KX16 and CDP-16KX16 core memories. Each technical manual is supported by a controlled drawing package that includes:

a. Schematics
b. Board assembly drawings
c. Theory of operation
d. Parts list

1.2.1.5 CDP-XI/PS Power Supply Technical Manuals

These manuals provide design detail, installation and maintenance information on the applicable system power supply. Each manual is supported by a controlled drawing package that includes:

a. Schematics
b. Theory of operation
c. Electronic assembly drawings
d. Mechanical assembly drawings
e. Parts list

1.2.1.6 Other Manuals

Other manuals associated with peripheral controllers and optional elements are provided as required.

Each manual generally consists of a technical manual and support drawing package, as described for other CDP-XI manuals.

1.2.2 Engineering Drawings

For maintenance purposes, this manual is supported by Drawing Package 21518004, which contains theory of operation, schematic diagrams, assembly drawings and other required engineering drawings. The drawing package is updated with the latest revision of each drawing. Since certain revisions may result in reassignment of drawing numbers, the text of this manual makes reference to drawings by title rather than by number.

For each contract, a custom CDP-XI Product Performance Specification is provided, defining the CDP-XI computer system physical, functional and performance parameters that will be met.

1.2.3 Abbreviations and Conventions

Table 1-1 lists the abbreviations found in this manual.

Conventions used in the text of this manual include:

a. Equipment panel nomenclature is reproduced in all uppercase characters.
b. The proper names of instructions, microcommands and signals are capitalized.
c. ZERO and ONE are used to indicate binary logic "0" and "1" states, respectively.
d. Octal numbers are preceded by a zero and hexadecimal numbers are preceded by a dollar sign for easy identification. Decimal and binary numbers are not prefixed.

Table 1-1. Abbreviations

Abbreviation	Meaning
Cal Data or CDP	California Data Processors
I/O	input/output
LFC	line-frequency clock
RAM	random-access memory
ROM	read-only memory
PROM	programmable read-only memory
MSI	medium-scale integration
LSI	large-scale integration
MMU	memory management unit
LED	light emitting diode
ACM	alterable control memory
DMA	direct memory access
CM	control memory
CC	microcommand location counter
CR	microcommand register
CS	control stack
SC	stack counter
LC	loop counter
MB	microbus
FR	file register
AB	A-operand bus
BB	B-operand bus
AU	arithmetic/logic unit
c	carry-out microcondition code
v	overflow microcondition code
z	zero data-value (microcondition) code
n	negative data-value (microcondition) code
p	positive data-value (microcondition) code
d	odd data-value (microcondition) code

Abbreviation	Meaning
MS	microstatus register
L	MS register link bit
V	MS register overflow bit
Z	MS register zero data-value bit
N	MS register negative data-value bit
P	MS register positive data-value bit
D	MS register odd data-value bit
AR	MACROBUS address register
AL	AR latch register
DR	MACROBUS data register
DL	DR latch register
PS	processor (macro)status register
LR	stack-limit register
RR	data-read register
IR	instruction register
XR	shift register
ER	emulate decode register
cps	characters per second
cpm	cards per minute
lpm	lines per minute
K	1,024 memory locations
max	maximum
A	Amperes
ac	alternating current
dc	direct durrent
V	Volts
ns	nanoseconds
Hz	Hertz
min	minimum

SECTION 2
PROCESSOR DESCRIPTION

2.1 OVERVIEW

The CDP-XI/00 (Figure 2-1) is a high-speed microprogrammed digital computer designed for application in a wide variety of computing and control applications. Microprogramming, combined with a powerful and flexible hardware architecture, permits the basic computer to be fully optimized to a specific application. The CDP-XI/00 is designed primarily for efficient, high-speed emulation of general-purpose computer architectures. It can also be applied as a direct function processor by implementation of problem-oriented microprograms.

2.1.1 System Organization

The overall CDP-XI system organization is shown in Figure 2-2. The CDP-XI system consists of a set of hardware and software elements that can be utilized in a wide variety of applications. A brief description of the elements of the computer system is given below. Details are given in later sections of this manual and in supporting manuals.

The main external data path of the system is a universal bidirectional bus called the MACROBUS. The MACROBUS is time-shared by all elements of the system, including the processor. Devices on the MACROBUS can communicate directly with other devices, independent of the processor.

The basic element of the system is the CDP-XI/00 processor, controlled by microprogram sequences (firmware) stored in the control memory. By changing the contents of the control memory, the entire operation of the system can be altered. An emulation system is implemented by placing the appropriate firmware in control memory, causing the processor to operate like the computer being emulated.

The CDP-XI/00 model designation refers to the basic processor without specific firmware in control memory.

The processor consists of three functional elements: control section, data section and I/O section. The control and data sections (referred to as the engine) contain the internal arithmetic/logic circuits, data paths, processor registers, control memory and timing circuitry of the machine. The I/O section contains the basic MACROBUS data, address and control circuitry for all parallel I/O operations in the system, plus an optional serial I/O communication channel (up to 4800 baud) for teleprinter or other compatible serial devices. Line-frequency clock and power-failure/restart circuits are also included as standard items.

A macropanel, representing the control panel of a general-purpose computer, is often provided in an emulation application. The macropanel is serviced by the processor as an I/O device interfacing with the MACROBUS. Special support firmware is normally provided for this unit.

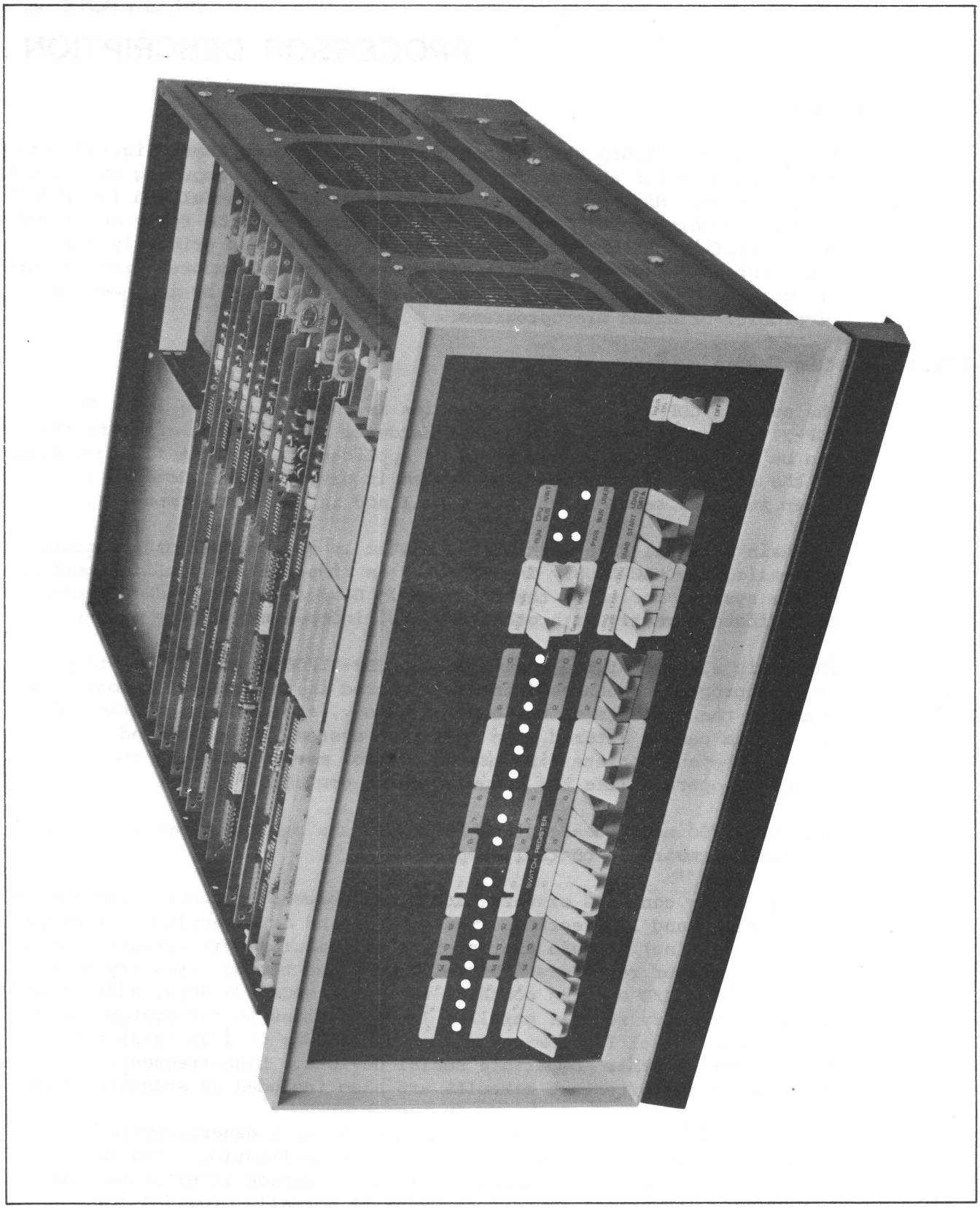


Figure 2-1. CDP-XI Computer System with CDP-XI/00 Processor and 80K Words of CDP-8KX16, 675-ns Core Memory Modules

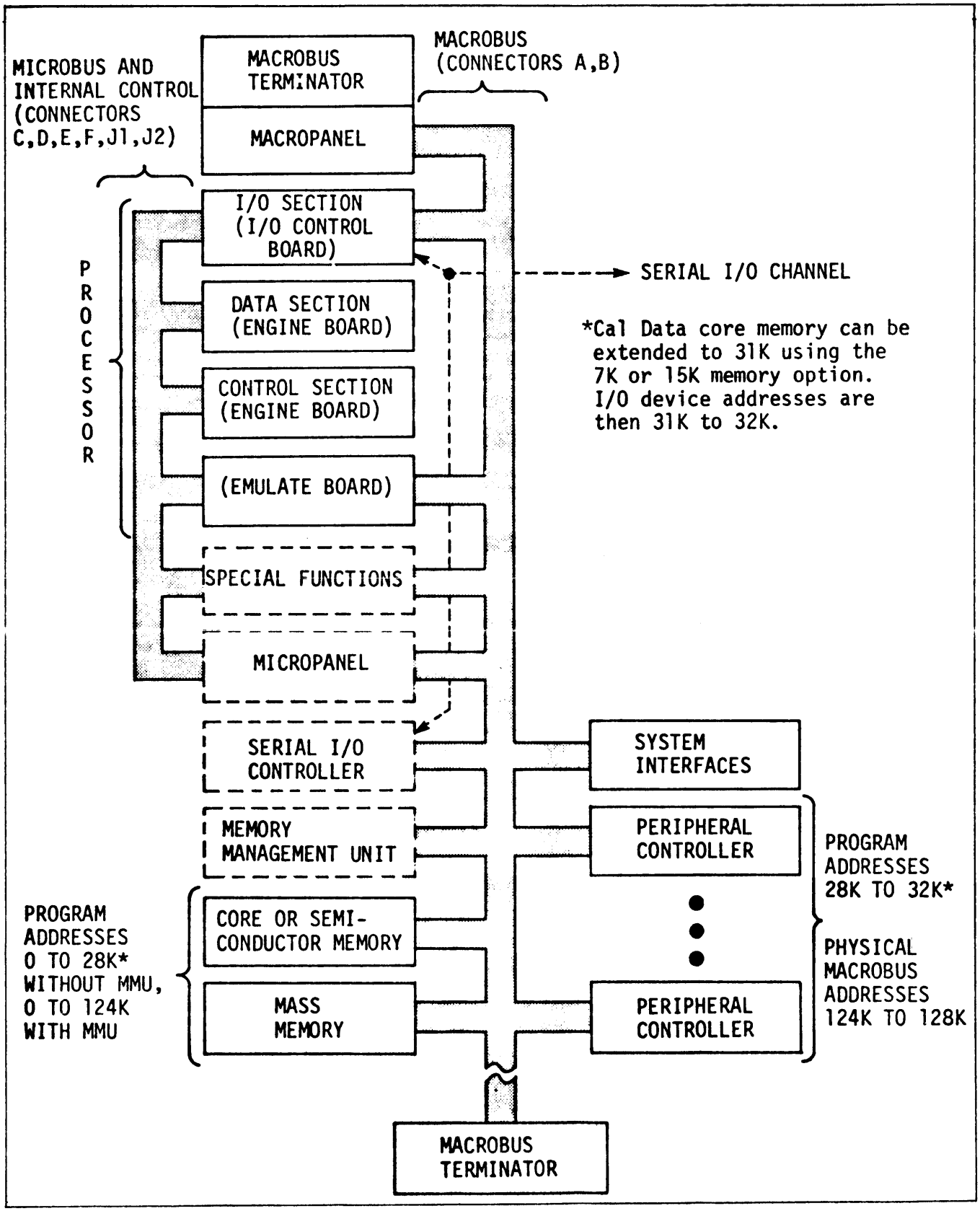


Figure 2-2. CDP-XI System Organization

A microcontrol panel is available to provide control and display for checking out and debugging firmware, and also for various maintenance and troubleshooting procedures. The micropanel consists of a plug-in interface board and a remotely mounted panel that permits the user to exercise direct control over the processor. Facilities are provided to construct full microcommands, to display microcommands as well as data and internal registers, and to execute microcommands on a single-step or trap-mode basis. The micropanel also contains either 16 or 32 words of ACM that can "overlay" the contents of control memory.

The micropanel can be used in conjunction with the macropanel and is useful for initial debugging of new firmware as well as for on-line troubleshooting of processor hardware, but is usually not required in an applied system configuration.

Cal Data core memory comprises modular blocks of 8K or 16K 16-bit words each contained on a single printed-circuit board. Each module plugs directly into the MACROBUS and is treated as an I/O device in the system. The maximum system capacity is 128K words. Two identical modules can be interleaved to achieve an increased effective throughput rate on the MACROBUS.

The MACROBUS can accommodate memory devices other than magnetic core, such as semiconductor ROM or RAM modules. The only requirement is that such units obey MACROBUS use rules. Modules of varying size and speed can be freely mixed with core memory. All MACROBUS devices can communicate directly with memory.

Peripheral device controllers and system interfaces are attached to the MACROBUS, as shown in Figure 2-1. The user can readily interface devices with the MACROBUS using simple design rules. Cal Data offers several standard peripheral subsystems to enhance system application. The subsystems offered to support normal programming and system development operations are:

a. Paper Tape Reader. High-speed photoelectric reader, 300 characters per second, fanfold tape.
b. Paper Tape Punch. High-speed punch, 75 characters per second, fanfold tape.
c. Card Reader. High-speed photoelectric card reader, 300 cards per minute with code conversion in the controller.
d. Line Printer. 80- or 132-column printer, 125 or 200 lines per minute.

2.1.1.1 Firmware Development Aids

Cal Data offers specialized hardware and software elements to aid users in developing custom firmware. These are briefly described below.

Alterable control memory (ACM) is a modular plug-in unit of the processor that contains increments of 256 words of electrically alterable control memory. The ACM also contains alterable elements for instruction emulation and decoding.

With the ACM, the programmer can load or read the contents of control memory directly and execute trial firmware code at normal processor

execution speeds. The ACM is particularly useful for dynamic system tests where external real-time events must be considered to fully evaluate a firmware microprogram. The ACM is supported by a software operating system that permits the programmer to use the system teleprinter to control the system.

The PROM programmer is a modular plug-in board that can be installed in a controller slot of the system chassis. It provides automatic inscriptions of code patterns into electrically alterable bipolar memory devices (PROMs). PROM devices inserted into sockets on the board are automatically inscribed and verified with code patterns stored in core memory. Operation is under control of the ACM software operating system. The PROM devices, once inscribed, can be switched into the system as control memory without removing them from the board, providing an immediate dynamic test of the firmware. The PROM programmer provides fast, accurate production of microprograms that have been generated and checked.

The following software is available to support CDP-XI firmware development:

a. Symbolic Microassembler. This program is a complete symbolic assembler that permits convenient coding and listing of microprograms. It is written in CDP-XI/35 emulator language and can be run on any CDP-XI or PDP-11 series computer having the required memory configuration.
b. ACM Software Operating System. This program is designed to provide operational control over execution of firmware in the ACM. It requires that the CDP-XI/35 emulator be resident in control memory.

2.1.2 Features

The CDP-XI/00 processor architecture combines general microprogramming capability with specialized optional features to permit high emulation speeds with efficient control-memory space utilization. The mechanical design used provides full modularity, mounting flexibility and service convenience. Cooling, power distribution and other critical system requirements are optimized for OEM applications. Conservative electrical implementation ensures wide margins, readily available components and reliable operation over a wide environmental range. Subassemblies are designed for easy assembly and automated testing, and the overall system is structured for simple, straightforward manufacturing procedures. Basic design features of the CDP-XI system are:

- 48-bit microcommand word length.
- Parallel execution of multiple functions per microcommand.
- 150-ns microcommand execution time.
- 16-bit data word length.
- 16 multipurpose file registers (16 bits each).
- Nine additional registers accessible by microcommand.
- 16-level hardware pushdown stack.
- Microcommand sequence repeat loop counter.
- Optional high-speed emulation instruction decode, function generation and interrupt-response hardware.
- Bit, byte and word manipulations.
- 256- to 4096-word control memory using bipolar ROM or PROM devices.

- Power-failure/restart circuitry included in the computer.
- Unique, control memory "overlay" provisions.
- Optional line-frequency clock.
- Optional Multiple, Divide, and single- and double-precision Shift microcommands.
- Hardware microprogram interrupts.

Memory and Input/Output

- 8K-word (675-ns cycle, 275-ns access) and 16K-word (850-ns cycle, 300-ns access) core memory modules.
- Interleaved data transfers between identical memory modules.
- Optional 7K or 15K extended addressing feature for memory expansion to 31K without memory management.
- Expansion to 124K or 127K of directly addressable memory with optional memory management unit.
- Universal asynchronous I/O channel (MACROBUS) with direct-memory-access capability.
- Four external priority interrupt levels.
- 16-bit parallel word or byte-mode transfers.
- Automatic MACROBUS delay timeout protection.
- Optional asynchronous serial I/O channel.

Microprogramming Aids

- Microcontrol panel.
- Alterable control memory and support software.
- PROM programmer.
- Symbolic microassembler.

Packaging, Power and Environmental

- 10-1/2 inch processor chassis with vertical board mounting from the top.
- Printed-circuit backplane with up to eight spare slots for memory and I/O controller boards.
- Four fans for high-volume, positive-pressure air flow through the chassis with provision for air filters.
- Modular power supply providing 36 A at +5 Vdc.
- Low-noise internal power distribution and grounding system.
- Convenient external I/O cabling.
- System designed to meet UL standards.
- 0 to 50^{o} C ambient operating temperature.
- 10 to 90% humidity (without condensation).

Electrical and Electronic

- Bipolar TTL integrated circuits (multisourced).
- Extensive use of MSI and LSI.
- Wide timing margins.
- Single-phase, square-wave clock.
- Conservative component derating.
- Metal-can transistors and hermetically-sealed passive devices only.
- High noise immunity I/O drivers and receivers.

2.2 FUNCTIONAL DESCRIPTION

A block diagram of the CDP-XI/00 processor is shown in Figure 2-3. The processor has three main functional sections: control, data and I/O. The control section contains the control memory and timing circuits that control the sequence of operations performed. The data section contains the arithmetic/logic, gating and busing elements that perform data transfers and manipulations. The control and data sections taken together are referred to as the engine. The I/O section controls the transfer of data, interrupts and status between the engine and external devices. The main communication path is the MACROBUS, which is a universal bidirectional 16-bit channel. All external devices, including memory (core or semiconductor), control panels and peripherals transfer data and control over the MACROBUS in a time-shared manner. Devices on the MACROBUS can communicate with the processor and directly with other devices, depending on the design. The I/O section can contain a full-duplex serial I/O data channel that communicates with conventional serial devices such as teleprinter, CRT units, etc. at rates up to 4800 baud.

A basic I/O device is the core memory, which is generally required in any system. Cal Data core memory modules are available in 8K- and 16K-word increments and can be added directly to the MACROBUS up to a typical maximum of 124K words*. By the use of the Cal Data extended addressing option, an additional 3K words of memory can be utilized in the system for a total capacity of 127K words. The last 4K (or 1K) addressable locations (out of a total of 128K) are reserved for I/O devices other than memory.

Semiconductor memory can be interchanged with core in any speed/capacity mix. The processor addresses memory locations like any other I/O devices.

Two types of control panel are available: a macropanel that is adapted to a particular emulation and permits the operator to control the system at the emulated level of operation, and a micropanel that permits control and display at the micro level and is useful for firmware development and hardware maintenance and troubleshooting. The macropanel is treated as an I/O device. Special interpretive firmware services the functions of the macropanel.

2.2.1 Control Section

A block diagram of the control section is shown in Figure 2-4. Control is organized around the control memory (CM), which stores the microprograms to be executed. Microcommands are 48 bits in length. CM capacity is from 256 to 4,096 words (48 bits each).

A 12-bit location counter (CC) addresses CM and advances on each clock step unless altered by a sequence change. Microcommands read from CM are held in a microcommand register (CR) during execution. The

*Maximum memory capacity of the basic system is 31K addressable words. A memory management unit is required for expansion beyond this capacity.

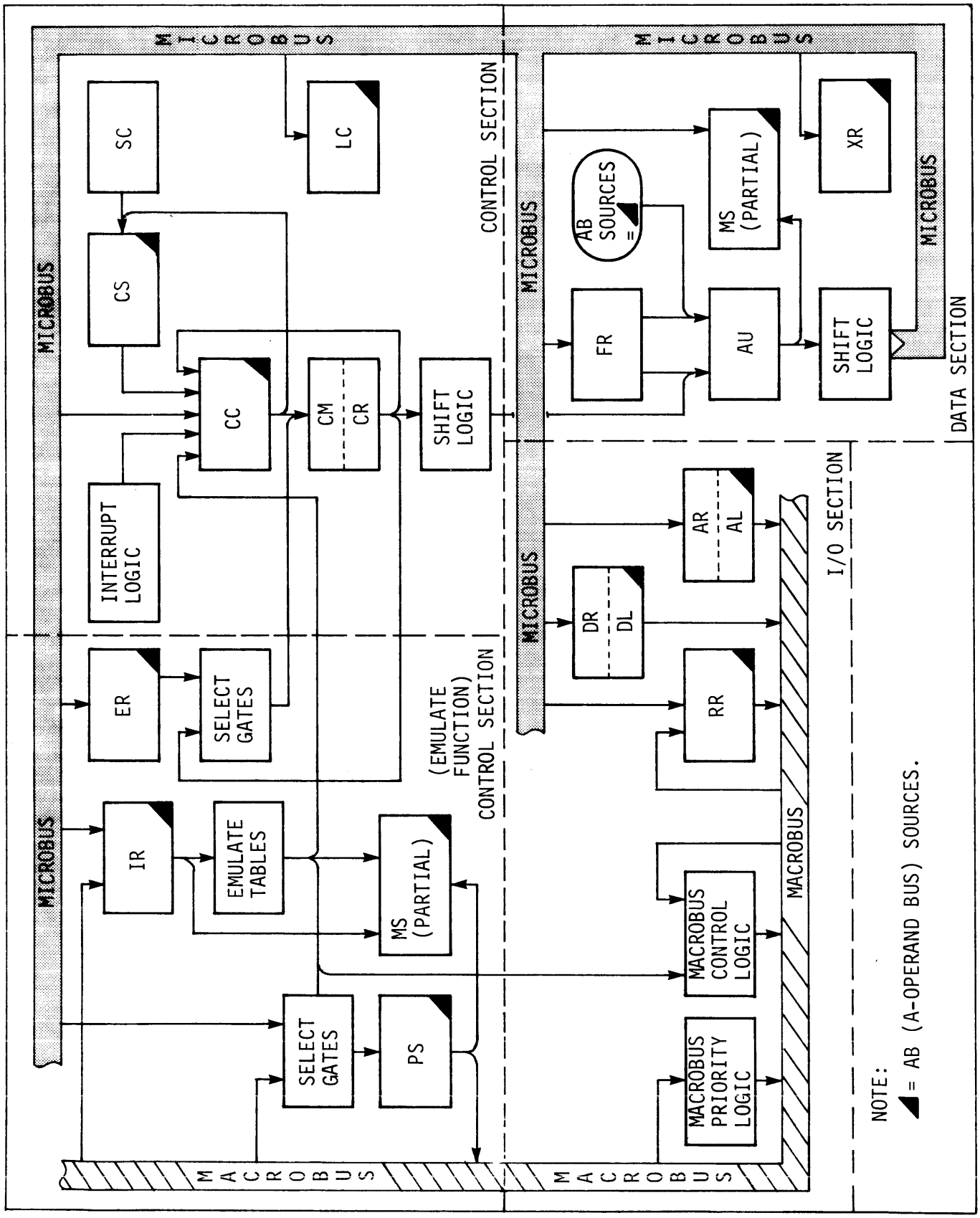
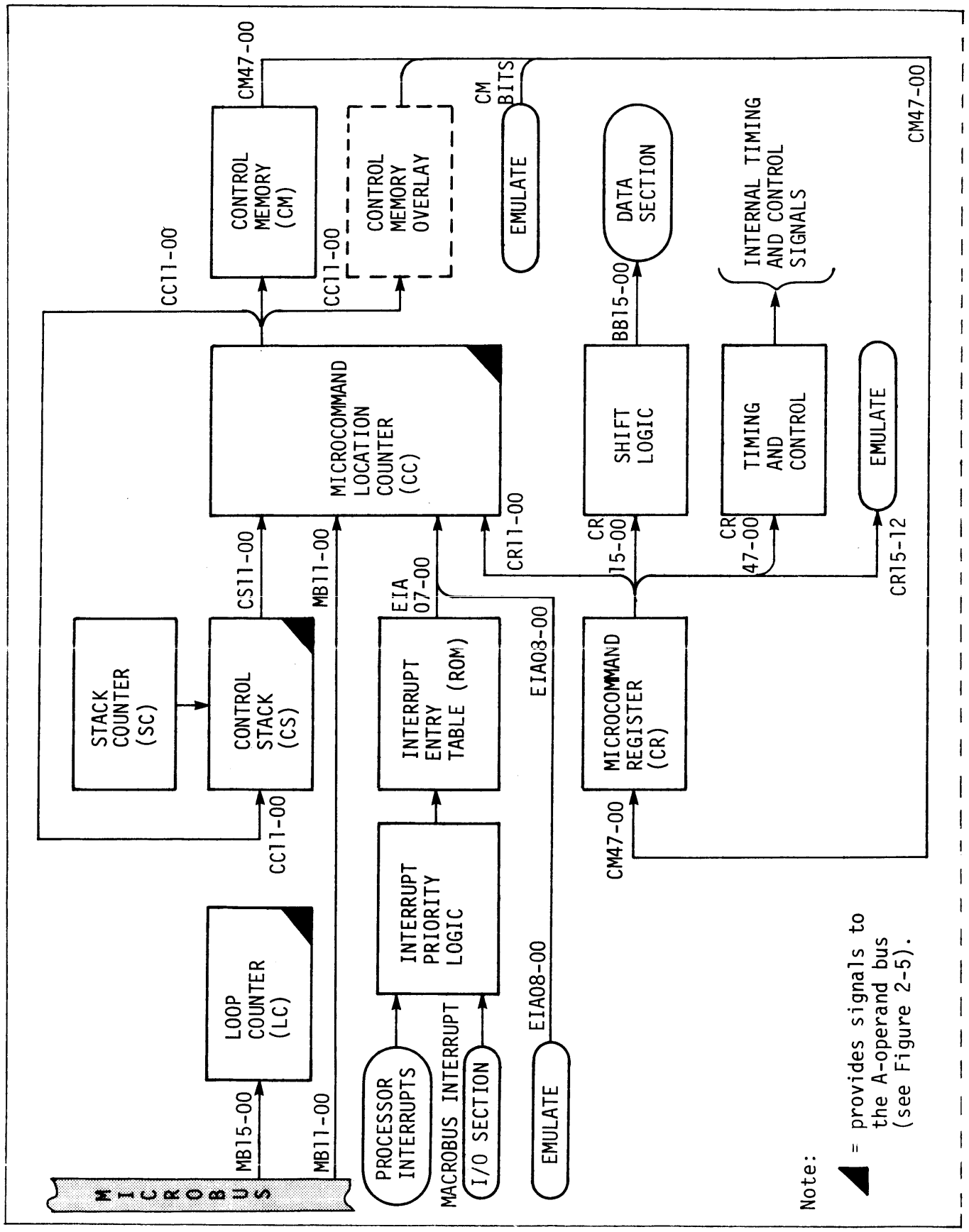


Figure 2-3. CDP-XI/00 Processor Block Diagram



(continued)

Figure 2-4. CDP-XI/00 Control Section Block Diagram

microcommands read from CM can be modified prior to input to CR for execution. Microcommands can also be entered manually into CR and executed from the micropanel.

A 16-level control stack (CS) is provided to permit the contents of CC to be saved and restored under microprogram control. This permits automatic nesting of microroutines and microprogram interrupts, giving increased speed and CM space efficiency. The system contains a unique CM overlay facility that permits designated areas of CM to be "patched" from auxiliary storage devices in the system or from the micropanel. This is a highly useful feature, since nonalterable storage elements are generally used to implement the CM.

An eight-bit loop counter (LC) is provided to permit single microcommands or entire strings to be repeated a specified number of times. This feature enhances execution speed of iterative loops.

Special features of the CDP-XI/00 are emulate, interrupt and instruction decode control circuits, located on a separate emulate board. These circuits provide:

a. Automatic table-generated addresses to CC to steer the microprogram directly to specific emulation microroutines, bypassing lengthy processing to decode instruction codes and addressing modes.
b. Automatic interrupt microroutine location entry into CC.
c. Automatic table-generated modifiers to microcommands read from CC.
d. Automatic modification of processor status conditions for the emulated instruction.
e. Direct designation of word or byte-mode operations.

Emulation related features and circuits are described in a separate emulation board technical manual written for each computer model.

2.2.1.1 Control Memory (CM)

The control memory is a high-speed, random-access unit. Three device implementations are used:

a. Read-only memory (ROM). These bipolar semiconductor devices are organized on chips of four by 256 (or four by 512) bits. Twelve such devices implement each 256-word (or 512-word) CM page. The code pattern in each chip is permanently inscribed during the factory manufacturing process and cannot be altered. ROM is used for high-volume production of fully debugged firmware.
b. Programmable read-only memory (PROM). These bipolar semiconductor devices are organized on chips of four by 256 bits, pin- and speed-compatible with the equivalent ROM. The code pattern in each device is electrically and permanently inscribed by a portable programming device. PROM is used for development and field debugging of firmware and also for low-volume production firmware packages.
c. Alterable control memory (ACM). The Cal Data ACM is a complete, modular control memory that can be installed in the CDP-XI in addition to or in place of ROM and PROM devices. It is implemented with bipolar random-access memory devices that can be electrically altered (read/write). When installed in the

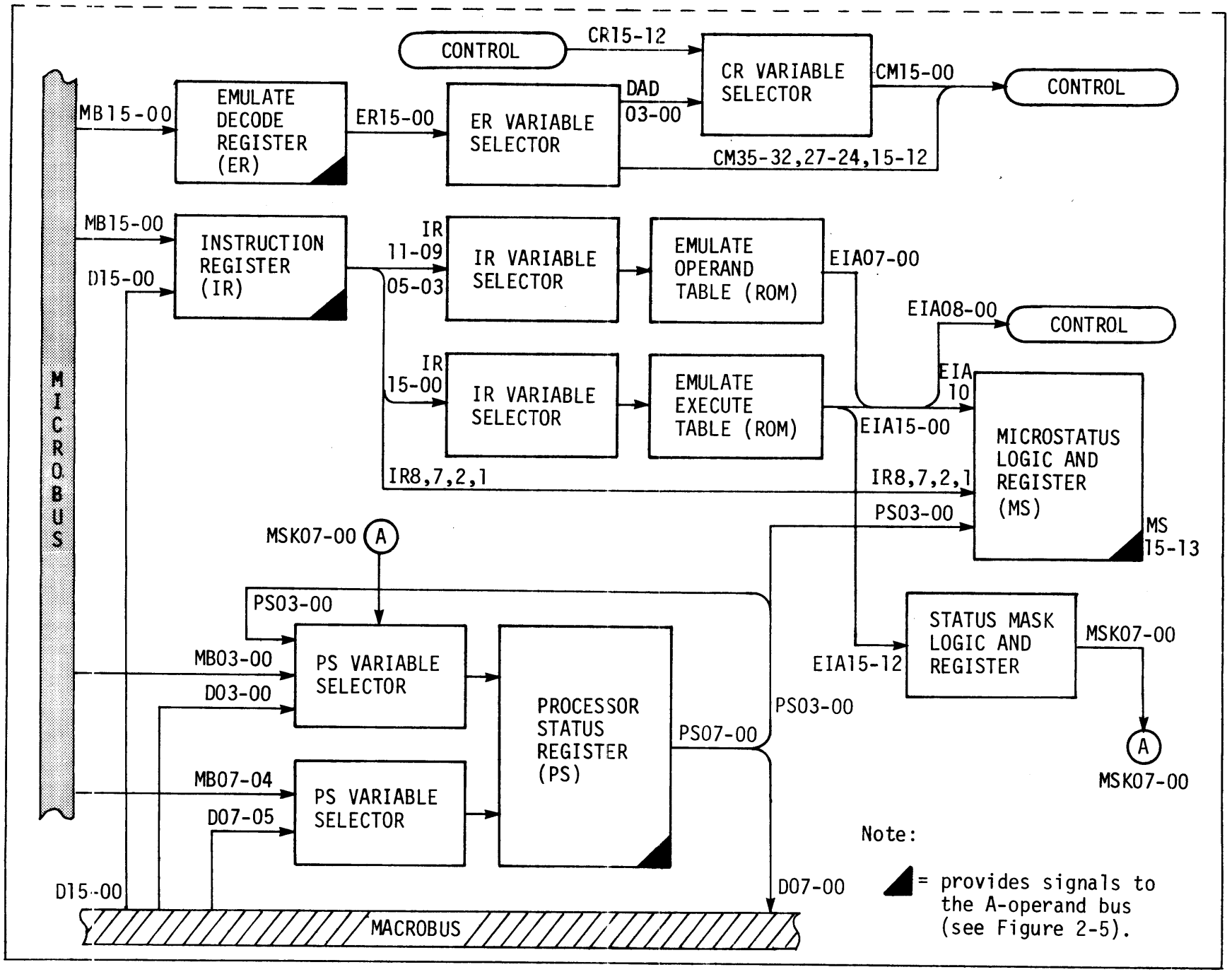


Figure 2-4. (Continued)

processor, ACM can be loaded and read via the MACROBUS using I/O microcommands. The ACM can then take control of the processor for execution of ACM firmware at real-time processor speeds. The ACM is most useful for initial and on-line check-out of new firmware prior to conversion to ROM or PROM devices.

The maximum capacity of CM is 4K words* when ROM or PROM devices are used. ACM capacity is limited to 1,536 words. Although each micro-command is 48 bits in length, the CM addressing structure of the micro-command limits direct access to 2K words; however, a paging scheme between 2K-word blocks permits convenient access anywhere within 4K words.

Control Memory Overlay. It is often desirable to alter the contents of CM, either temporarily or permanently. When nonalterable devices are used, the usual requirement is replacement of the existing devices. The CDP-XI/00 incorporates circuitry that permits either one or two 32-word blocks of auxiliary memory to functionally replace designated 32-word blocks in CM. This enables "patching" for corrections, additions or deletions from existing firmware, temporary overlap for diagnostic and troubleshooting operations, etc.

2.2.1.2 Location Counter (CC)

The location counter is a 12-bit binary counter/register that points to the location in CM of the next microcommand to be executed. The microprogram sequence can be altered conditionally or unconditionally as specified by the programmer and the state of the system. A sequence change is made by loading CC from one of the following sources:

a. CR for programmed branches.
b. Microbus, for computed branches.
c. The current CS register.
d. A vector from the emulate table.
e. An interrupt vector.

CC normally advances sequentially to the next location through all 4K locations in CM, including the wrap-around transition from 4,095 to 0, unless the normal sequence is altered.

CC modifiers from CR and the emulate table are 11 bits long, permitting branches to occur from these sources within only a 2K-word area. The high-order bit of CC is unaltered for such branches. To branch to a location outside a 2K-word area, the programmer must execute a micro-command that transfers a full 12-bit branch address via the microbus (MB). Interrupt vectors are to only the first 256 CM locations (i.e., the four most-significant CC bits are forced to ZERO).

Certain conditions cause an automatic reset of CC to location zero (a corresponding status bit is set for each condition):

a. Access to a nonimplemented or unused area of CM.
b. A catastrophic system error.
c. A power-up sequence.

*Auxiliary power is required above 512 words.

The contents of CC can be read by microcommand via the A operand bus (AB). For systems that do not contain an implemented CS, this provides a means of saving a return location in control memory.

2.2.1.3 Microcommand Register (CR)

The 48-bit CR stores the current microcommand read from CM for execution. The microcommand from CM can be modified prior to entry into CR by a function specified by the special decode circuitry on the emulate board. CR can also be loaded from the micropanel to permit direct operator control of internal functions. The low-order 11 bits of CR modify CC when a branch operation is specified by the microcommand in CR.

Microcommand Sequencing and Timing. The basic clock cycle is 150 ns and, ordinarily, a microcommand is read from CM and executed on each cycle. There is a one-clock delay between the time CC addresses a word in CM and the time that the microcommand is transferred to CR for execution. For this reason, when the normal CC counting sequence is modified, two clock cycles are required to access the microcommand at the branch location and transfer it to CR. Furthermore, the microcommand accessed at the time CC is modified is transferred to CR even though a branch is being made. Whether or not this "extra" microcommand is executed can be specified by the programmer. The following sequence illustrates the operation:

Time	CC	CR	Operation
T-1	X	(X-1)	
T	X+1	(X)	Branch to Y specified.
T+1	Y	(X+1)	Microcommand at X+1 can be executed.
T+2	Y+1	(Y)	Microcommand at branch location.

In addition to sequence modification, the programmer can specify that the following microcommand be skipped. In this case, the following microcommand is transferred to CC, but execution is inhibited. This action is not considered to be a sequence change since CC continues normal sequential counting.

2.2.1.4 Control Stack (CS)

CS contains 16 12-bit registers that are accessed via the four-bit up/down stack counter (SC). When a CC-save is specified by a microcommand, the contents of CC are transferred to CS. The contents of CC are always one greater than the location of the microcommand specifying the save. Likewise, a microcommand can specify a return operation that transfers the contents of the current CS location to CC. The return microcommand can simultaneously transfer the (incremented) contents of CC to the CS register that contained the return address. Incrementing and decrementing SC can be specified independently of the save and return functions. CS permits convenient implementation of re-entrant and multilevel subroutes at the micro level. Any microcommand branch condition can specify a save operation with an automatic return to the calling sequence using a return microcommand.

SC counts up from 0, modulo 16, and rolls over the boundary in either direction. There is no indication given for a stack overflow. It is the programmer's responsibility to maintain the stack within limits.

The contents (current location) of CS can be read by microcommand; however, CS cannot be directly loaded and SC is not directly accessible to the microprogram. The contents of CS, therefore, cannot be saved in the event of a power interruption. It is mandatory that provision be made to execute all returns in CS within the time available for power interruption. Since several milliseconds are available, this imposes no practical restriction on the use of the stack.

2.2.1.5 Loop Counter (LC)

A powerful feature of the CDP-XI/00 is the eight-bit LC that permits a single microcommand or a group of microcommands to be automatically repeated up to 256 times. LC is loaded via MB and can be read with a microcommand. In a repeat sequence, LC can be tested for a zero or nonzero condition by any microcommand in the sequence, with a branch operation executed if the condition is met. LC is decremented each time it is tested. Individual microcommands can also be repeated the number of times specified by LC.

2.2.2 Data Section

A block diagram of the Data Section is shown in Figure 2-5. The data section contains the basic arithmetic, logic and busing elements of the processor required for manipulation and transfer of operands and status information.

The data section utilizes 16-bit parallel data paths and operational elements. Provision is made for byte-mode operations. The general file-register (FR) structure provides either eight or 16 general-purpose registers directly addressable by each microcommand. The output of any file register can be selected as either the A- or B-operand input to the arithmetic/logic unit (AU), and the results of the operation are routed via MB to many destinations (including FR) within the processor.

Dynamic condition codes indicating conditions of the operational results (e.g., overflow, negative, etc.) are generated for each microcommand executed. These conditions can be saved as static status bits. Either the static or the current dynamic conditions can be tested by any microcommand.

2.2.2.1 File Registers (FR)

The file registers provide general-purpose storage within the data section. Either eight or 16 registers of 16 bits each can be implemented.

The file registers permit the following simultaneous operations to be performed:

a. Any two file registers can be specified as the A- and B-operand sources to AU.

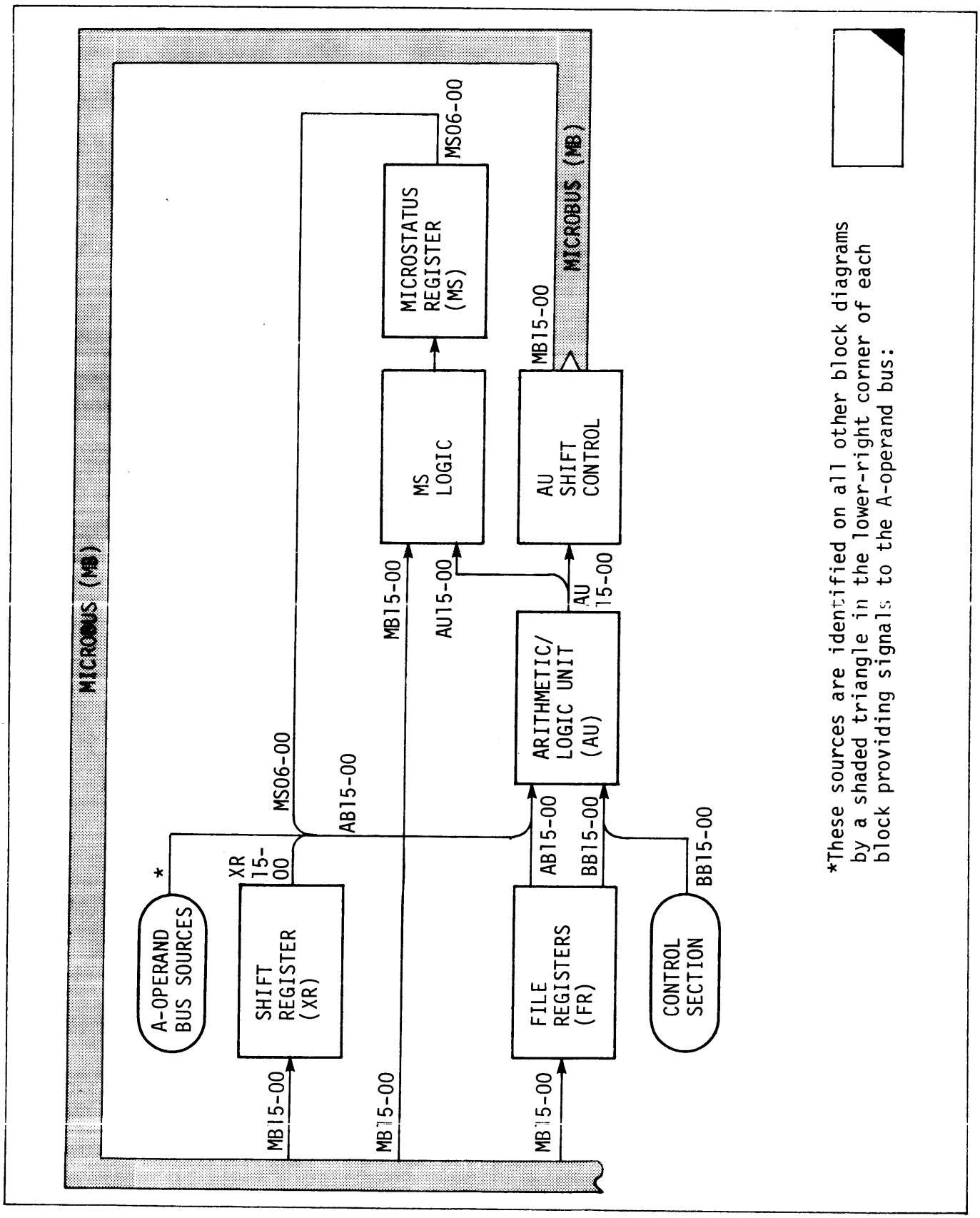


Figure 2-5. CDP-XI/00 Data Section Block Diagram

b. The file register selected as the A-operand source can also be specified as the destination register.
c. Any file register can be specified as a destination register for MB.

2.2.2.2 Operand Buses (AB, BB)

Operands are transferred to AU via the A-operand bus (AB) and B-operand bus (BB). All microcommands executed by the processor involve the use of information on one or both of these buses.

AB sources can be selected from any one of the file registers or from one of 11 other operational registers in the processor. There are five unused AB source addresses, of which two are designated for user-defined functions.

The BB source can be either:
a. Any one of the file registers.
b. The least-significant 16 bits of the current microcommand contained in CR.

The second source listed above represents a 16-bit literal value contained in the microcommand.

2.2.2.3 Arithmetic/Logic Unit (AU)

AU is a 16-bit parallel element that performs arithmetic (Appendix A) and logic functions on two variables input via AB and BB with the link (L) status bit from the microstatus register (MS) used as a carry input for addition and subtraction operations. A carry output (c), resulting from AU operations, can be tested as a conditional skip or branch condition and can also be stored (in the L bit) as a static status condition in MS.

Each microcommand specifies, either implicitly or explicitly, the AU operation to be performed and the use of the L input. A total of 15 logical and eight arithmetic functions are implemented.

2.2.2.4 AU Shift Elements

A separate set of gates is provided for shifting an AU operand. The following can be performed:
a. Left shift one bit.
b. Right shift one bit.
c. Shift eight bits (swap more-significant and less-significant bytes).

For shift operations, the L bit in MS is normally used as the shift carry-in and c is the bit shifted out of the shift gates. This carry bit can be saved as L for the next AU operation.

Provision is made for both single- and double-length shifts, either of which can be logically open, closed or arithmetic. Double-length shifts are performed in conjunction with the MACROBUS data register (DR), which is a 16-bit shift register. In this case, the L input and c output are dependent on the direction of the shift.

Shifts are performed by using shift operation codes in microcommands. Therefore, an AU operation is not performed simultaneously with a shift operation, because the A operand is always used in the shift. Shift microcommands must specify the type of shift to be performed and the carry input function.

Multibit shifts can be performed by the use of LC by setting up a shift count and repeating the microcommand. This permits execution of shifts of all types to be performed in one clock step per bit shifted.

2.2.2.5 Microbus (MB)

MB receives the resultant output from an AU or shift operation and provides the transfer path to all internal processor destinations. Each microcommand specifies a destination address to one MB location. In addition, by setting one bit of the microcommand, the AU result can be transferred to the A-operand source.

2.2.2.6 Microcondition Codes

For each operation performed by the AU or shift gates, a set of condition codes is dynamically generated, describing the result. These are:

a. Carry-out (c). The carry-out is generated as the arithmetic carry for an add operation, the borrow for a subtract operation or the shift carry-out for a shift operation.
b. Overflow (v). Overflow is generated for add, subtract or shift operations. The conditions under which overflow occurs depends on the operation.
c. Zero (z). The zero condition exists when all bits of the result are ZERO.
d. Negative (n). The negative condition exists when the most-significant bit of the result (shifted, if applicable) is ONE.
e. Positive (p). The positive condition exists when the result is greater than zero (not zero and not negative).
f. Odd (d). The odd condition exists when the least-significant bit of the result is ONE.

The last four conditions are referred to as data value codes and are generated from the value of the AU result on MB.

A microcommand can specify dynamic conditional testing of the microconditon codes generated as the result of an operation, and the conditional test can cause a skip of the next microcommand or a branch to a new microprogram location. This capability saves considerable time over machine designs that require conditional testing to be performed on the condition generated by a *previous* operation.

2.2.2.7 Microstatus Register (MS)

The six dynamic condition codes can be saved as static microstatus bits in MS. Each microcommand can specify separate storing of the carry/overflow and the four data value codes in MS. These static microstatus conditions can then be tested (instead of the dynamic microcondition codes) for conditional skips or branches, by microcommands.

MS is 16 bits in length. In addition to the six microcondition codes, other status bits are stored in this register. The contents of MS can be read via AB and can be loaded as a destination via MB. The complete set of status bits contained in MS is defined in Table 2-1.

2.2.2.8 Word and Byte Operations

The AU and shift elements of the processor handle 16 bits and, therefore, execute full word operations. The processor is also designed to operate on bytes (half words), if so specified by a microcommand. The byte mode can be designated as unconditional or conditional. In the conditional case, a byte-mode operation is performed only if the emulation circuitry indicates that the instruction being emulated is a byte-mode instruction.

The system has the capability of transferring either words or bytes via the MACROBUS. For arithmetic and logic byte operations involving AU, the specified operation is performed on the full 16-bit A and B operands. Since microcondition codes are generated on only the less-significant byte (bits 07 to 00) of the result, the bytes to be manipulated must be right-justified. Carry bits propagated out of the less-significant byte affect the results in the upper byte. For example, consider addition of the following two right-justified bytes:

```
   A : 00000000 10110110   (-74)
  +B : 00000000 11101011   (-21)
      (00000001)10100001   (-95)
```

Microcondition codes are generated from the *less-significant* byte as follows:

C = 1, v = 0, z = 0, n = 1, p = 0, d = 1.

The result for byte-mode operations is interpreted for the *less-significant* byte only. In many cases, it is desirable to extend the sign of the *less-significant* byte across the entire word (e.g., where word and byte arithmetic operations are mixed). A microcommand is provided that will insert the state of the microstatus L bit into the upper eight bits of a word. Thus, if the state of the c bit from the previous example is saved as L, the sign extended result is:

1111111101000001

This can be generated by execution of the Sign Extend microcommand.

For byte shift operations, shifting is performed on only the *less-significant* byte. The more-significant byte remains unchanged. The carry input and microcondition codes are associated with the *less-significant* byte. Examples are:

a. Byte mode, open left shift:

```
A = 00000011   10101101 <-- L
R = 00000011   0101101L
       (c)1 <--+
```

c = 1, v = 1, z = 0, n = 0, p = 1, d = L.

Note that L is the shift carry input and that the carry-out is the most-significant bit of the lower byte.

Table 2-1. Microstatus Register Bit Definitions

MS Bit	Symbol	Name	Description
00	L	Link	Stored state of dynamic carry-out (c) of AU or shift gates.
01	V	Overflow	Stored state of dynamic arithmetic or shift overflow (v).
02	Z	Zero	Stored state of zero (z) data value code.
03	N	Negative	Stored state of negative (n) data value code.
04	P	Positive	Stored state of positive (p) data value code.
05	D	Odd	Stored state of odd (d) data value code.
06		CM boundary	Indicates that CC accessed an unused word location in an implemented page of CM.
07		Halt	Indicates that the ENAB/HALT switch on the macropanel is set to HALT.
08 to 15			Unused.

b. Byte mode, open right shift:

A = 00000011 L ⟶ 10101111
R = 00000011 L1010111 ⟶ 1

$c = 1,\ v = \overline{L},\ z = 0,\ n = L,\ p = \overline{L},\ d = 1.$

The CDP-XI/00 has an extensive complement of shift microcommands that include arithmetic as well as logical open and closed forms, both single- and double-precision (double-length shifts involve DR).

2.2.3 Input/Output Section

A block diagram of the I/O section is shown in Figure 2-6. The I/O section controls transfer of data and control signals between the processor, and the external MACROBUS (and serial I/O channel).

The MACROBUS (Section 3) is the standard I/O channel provided with the CDP-XI/00. Because the CDP-XI is designed for broad applications as an efficient emulator of existing computer architectures, it may be desirable to duplicate the hardware I/O system of the subject computer as

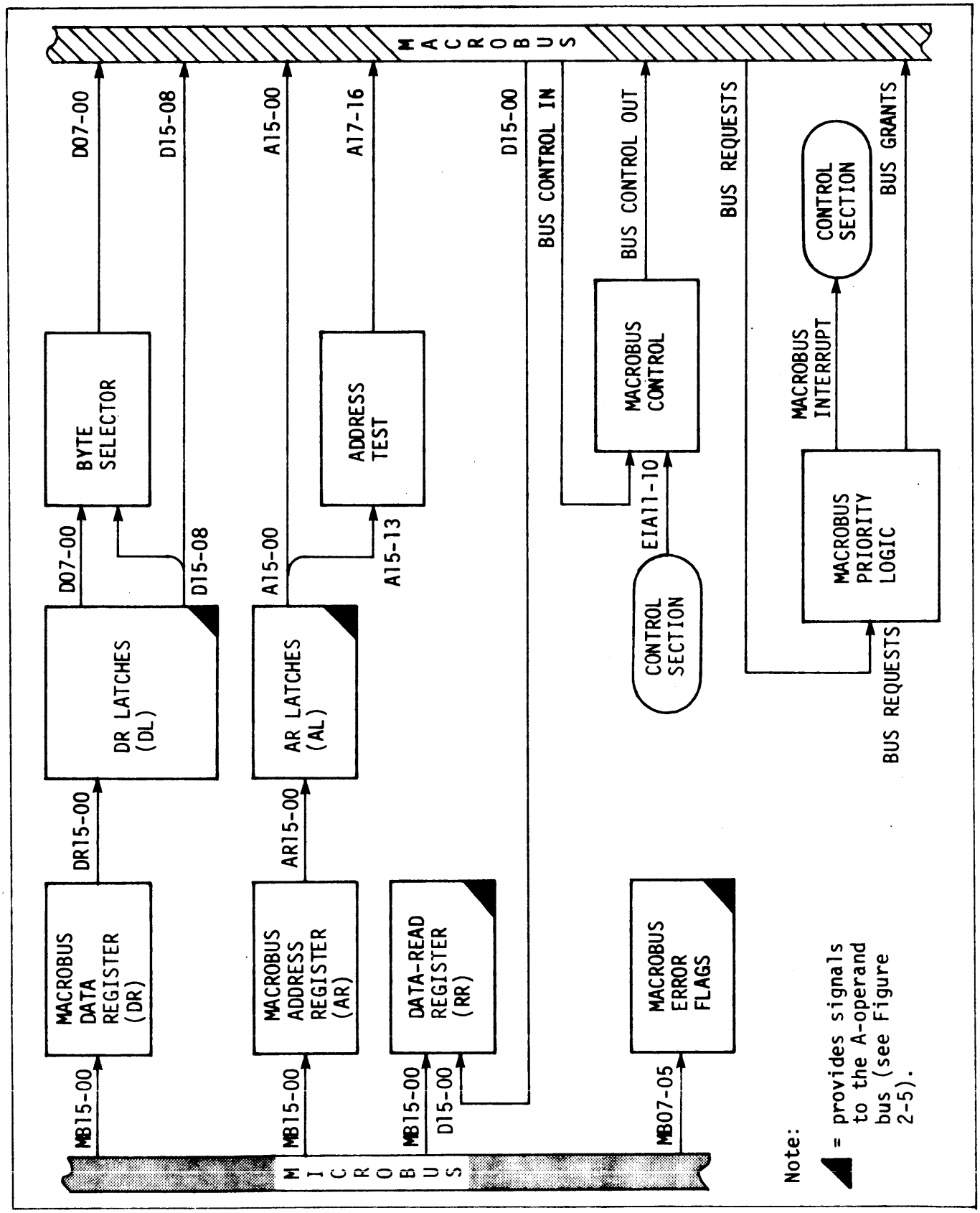


Figure 2-6. CDP-XI/00 I/O Section Block Diagram

well as to emulate the internal instruction repertoire. Because of the modular structure of the CDP-XI/00, such a modification does not require redesign of the engine.

The I/O section consists of internal registers and control circuitry directly accessible by the microprogram. In addition, MACROBUS priority control is provided to handle bus access priority requests. Operation of the MACROBUS priority control is independent of the processor.

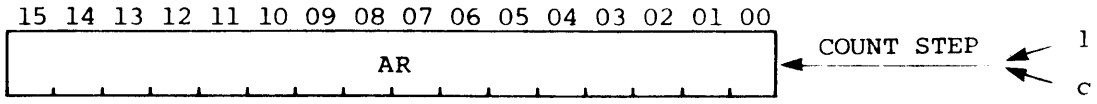
2.2.3.1 MACROBUS Address Register (AR)

The 16-bit AR stores address words needed for MACROBUS operations. Associated with AR is a 16-bit latch register (AL) that is loaded with the contents of AR at the time a MACROBUS write operation is initiated by a microcommand. The address is held in AL during the entire MACROBUS cycle. AR can be loaded with a new value at any time after the MACROBUS cycle is initiated. AR is loaded by microcommand via MB.

The 16-bit output of AL becomes an 18-bit physical address to the MACROBUS. The most-significant two bits of the 18-bit physical address are ZERO when the AL address is in the range 0 to 28K. For extended addressing beyond 32K (28K memory and 4K I/O addresses), the memory management unit is required.

The three most-significant bits of the 16-bit AL address are monitored by address test circuits to detect an I/O address in the range 28K to 32K, which are usually assigned to I/O devices. When any address within this series is detected, the two most-significant bits of the 18-bit MACROBUS address are set to generate a physical address in the range 124K to 128K.

AR has counting capability and can be used as an up-counter under microprogram control. The count can be by one or by one conditioned on the state of the carry microcondition code (c) generated by the AU or shift operation (the microcommand designates the count input to be used). This is illustrated below:



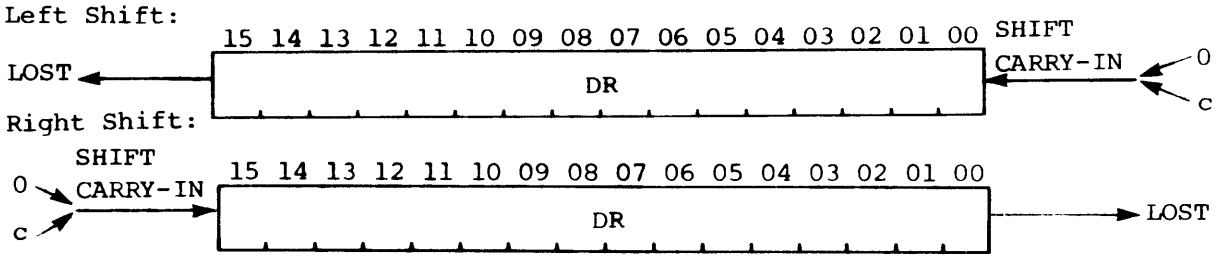
2.2.3.2 MACROBUS Output Data Register (DR)

The 16-bit DR stores data words (or bytes) needed for MACROBUS operations. Associated with DR is a 16-bit latch register (DL) that is loaded with the contents of DR at the time a MACROBUS write operation is initiated by a microcommand. The output word is held during the entire MACROBUS cycle. DR can be loaded with a new value at any time after the MACROBUS cycle is initiated. DR is loaded by microcommand via MB.

The output of DL is transferred to the MACROBUS through a set of gates (multiplexer) that also performs any necessary output byte conversions.

Other elements of the I/O section can also transmit output words to the MACROBUS. These are the processor status register (PS), serial I/O channel, line-frequency clock (LF) and stack-limit register (LR). These elements are handled separately from normal processor output data transfers and are not described here.

DR can shift both left and right. It is used in conjunction with AU shift elements to implement double-length shift microcommands. Additionally, DR can be used under microprogram control for shift operations independently of AU shift elements. Microcommands can specify the direction of shift and the shift carry input. The latter can be ZERO or the state of the carry microcondition code generated by either the AU or shift operation executed by the microcommand that specifies the DR shift. This is illustrated below:



The shift carry-out is lost, since it cannot be saved or tested. There is no provision for generating other dynamic condition codes while the shift is in progress. Static data-value status, however, can be generated by using another microcommand to transfer the result in DR through AU, where the test is performed. Multibit shifts can be executed in DR by the use of LC to repeat the shift microcommand.

2.2.3.3 MACROBUS Input Data Registers (RR and IR)

Data inputs from the MACROBUS can be routed under microprogram control to one of two registers: the data-read register (RR) or the instruction register (IR). The microcommand specifying the read operation also specifies which of these two registers receives the input word.

RR is generally used to receive data words and operands. This 16-bit register receives inputs from the MACROBUS via a set of gates that performs any necessary input byte conversions. RR can also be loaded directly via MB under microprogram control.

IR holds an instruction during emulation and is associated with the special circuitry on the emulate board.

2.2.3.4 MACROBUS Priority Control

MACROBUS priority control circuits monitor the bus requests made by the processor and external devices, establish the priority of MACROBUS access, and issue bus-mastership grant and synchronization signals to requesting devices. Processor interrupt requests are also handled by the MACROBUS priority control circuits, which transmit these requests to the processor interrupt control on the emulate board.

2.3 PHYSICAL DESCRIPTION

All CDP-XI/00 processor and system elements are modular and can be mounted in a standard CDP-XI processor chassis (Figure 2-7) that occupies 10.5 inches (26.7 cm) of a 19-inch (48 cm) RETMA rack. This modularity gives the user maximum flexibility in system design and configuration.

The standard CDP-XI processor chassis dimensions are:

10.43 inches high.
19.00 inches wide.
24.00 inches deep.

Hardware items included with the standard processor chassis are:

a. Chassis box.
b. Top and bottom covers.
c. Hinged fan panel and four fans.
d. Chassis slides.
e. Macropanel bezel and overlay.

A power supply mounts at the rear of the chassis. The ac power cord exists via the rear from a power-supply control panel accessible at the rear of the chassis. This panel also has the ac line switch, fuses, convenience outlet (115 Vac model only) and macropanel lock switch.

The four fans provide horizontal, positive-pressure air flow across the vertical processor boards and power supply. The panel is hinged to permit moving the fans when boards are removed or installed.

System electronics are mounted on modular printed-circuit boards that plug vertically into a printed-circuit backplane. The boards are 8.94 by 15.7 inches (22.71 by 39.9 cm), and are physically compatible with PDP-11 series boards.

The macropanel (Section 4) is mounted on a printed circuit board that plugs into the first connector row of the backplane. The macropanel is covered by an overlay held in place by the bezel. The bezel and overlay are removable from the front when the chassis is installed in a rack.

2.3.1 Internal Arrangement

Figure 2-8 shows a top view outline of the chassis and backplane configuration. The connector pattern consists of six columns, designated A to F. The internal MACROBUS is routed in columns A and B, with all interface connections made directly at these points. All signal interconnection on the backplane is via point-to-point etched lines. Power and system ground connections from the power supply are made directly to the backplane.

The basic CDP-XI system is provided with the processor backplane, mounted at the front of the chassis, and contains thirteen connector rows. The backplane has five closely-spaced (0.75 inch or 1.90 cm) rows of connectors for the processor and macropanel, and seven widely-spaced (0.9 inch or 2.29 cm) rows of connectors for the installation

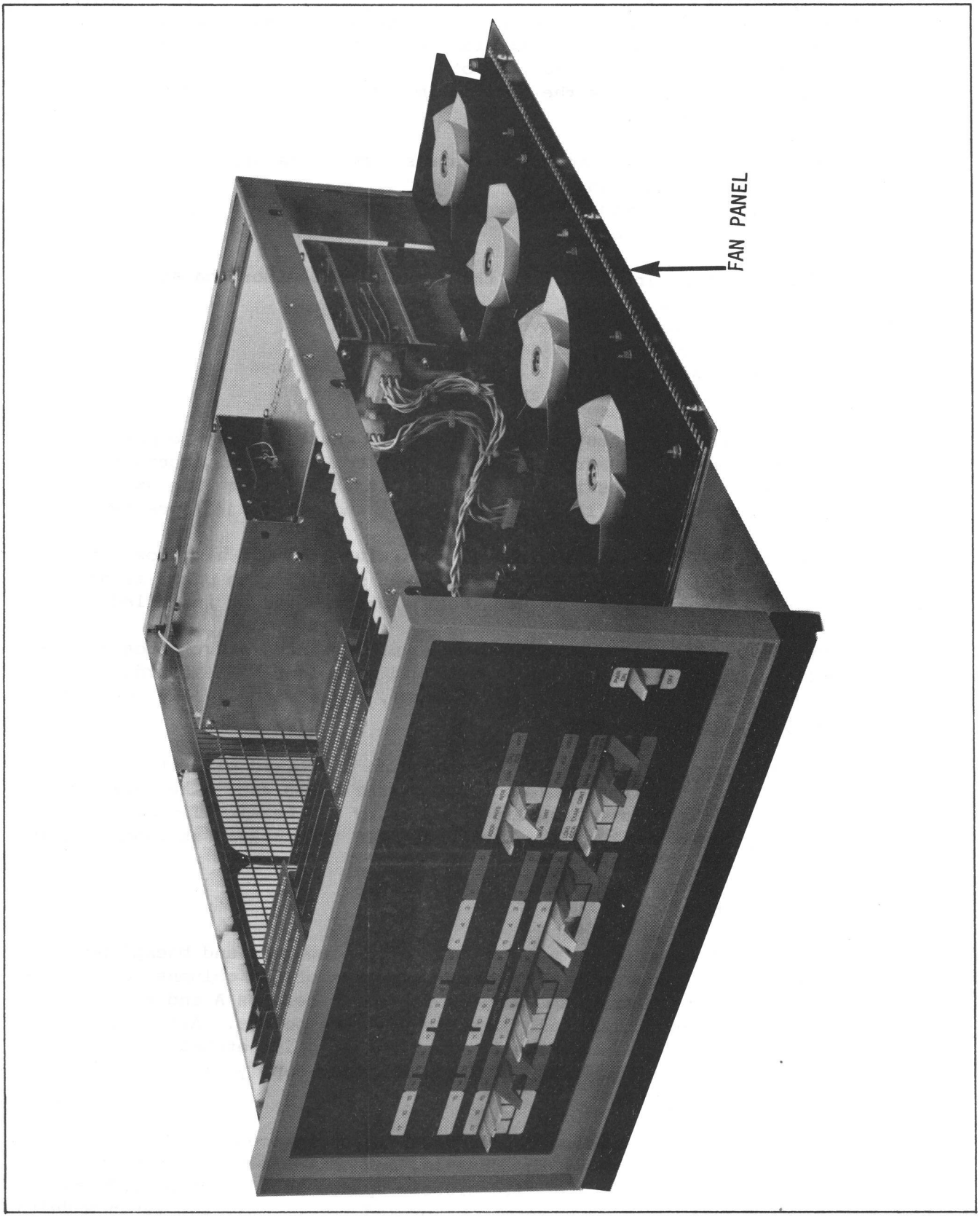


Figure 2-7. Standard CDP-XI Chassis with Processor and Macropanel Installed (Fan Panel is Shown Down)

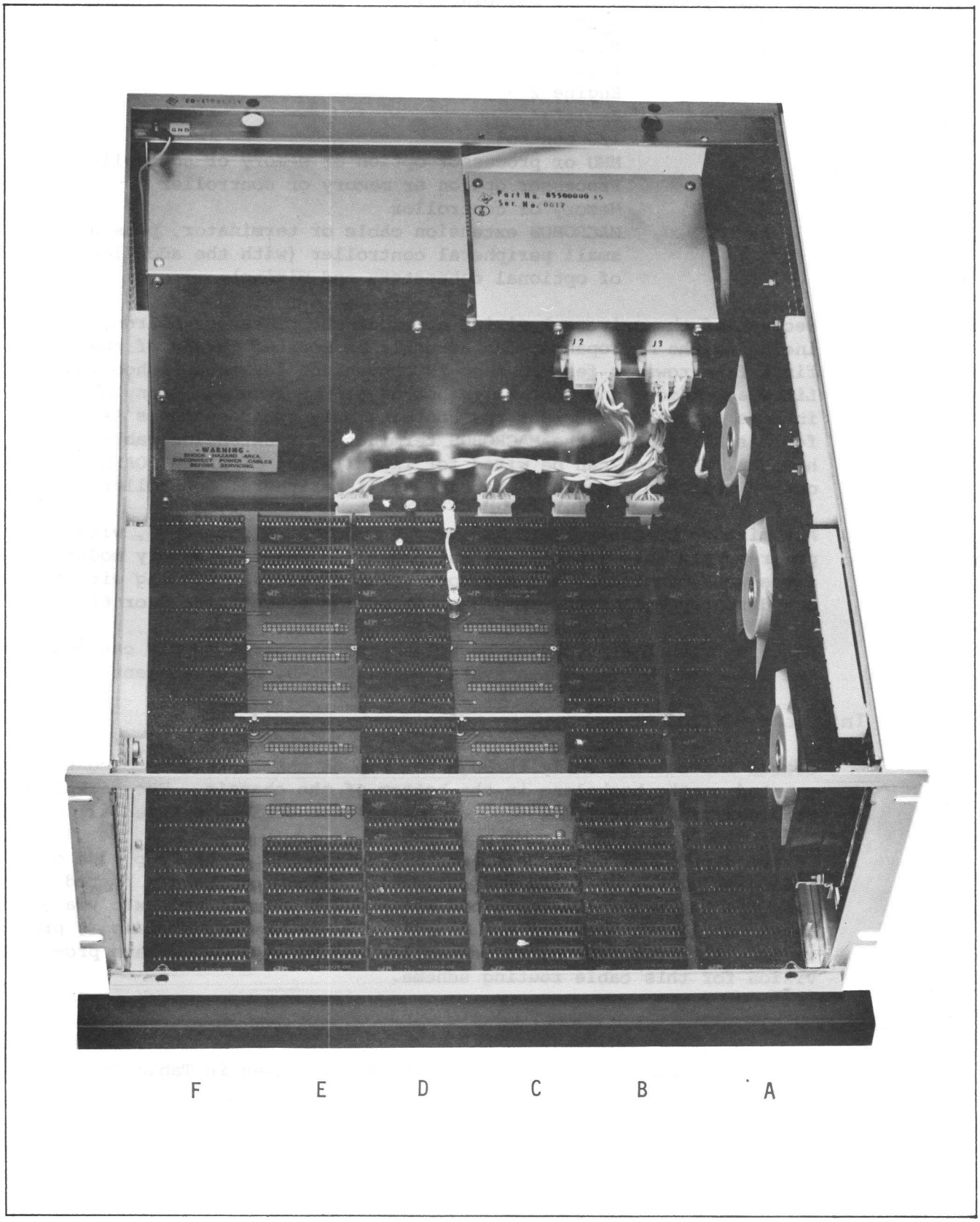


Figure 2-8. CDP-XI Processor Chassis, with 12-Slot Backplane

of options, memory modules and I/O controllers. The usual board slot allocation in this backplane is:

Row	Board
1	Macropanel
2	Engine 1
3	Engine 2
4	Emulate
5	I/O control
6	MMU or processor option or memory or controller
7	Processor option or memory or controller
8 to 12	Memory or controller
13	MACROBUS extension cable or terminator, plus a small peripheral controller (with the addition of optional connectors and wiring).

Because of the universal wiring arrangement, processor boards, including the macropanel, can actually be mounted in any of the first five rows, a feature that is convenient for troubleshooting. Likewise, memory and controller boards can be located in any order in rows 6 to 12. The last row contains only two connectors (A and B) for installation of the MACROBUS terminator or extension cable; however, the slot can be wired for installation of a QUADBOARD™ option that can contain a standard small peripheral controller.

Any mixture of controllers and memory modules is permitted, within power-supply load limits. In the processor chassis, memory modules are always mounted in a contiguous group of slots beginning with the one directly behind the processor or option board (slot 5 or 6).

Space between the processor backplane and the power supply can contain one standard PDP-11 series system unit, DD11-B, or equivalent.

2.3.2 Interface Wiring

Boards insert vertically through the top of the chassis into connectors mounted on the backplane in the bottom of the chassis. The backplane provides printed-circuit (and wire-wrap) connections between all boards.

Device controller cables are generally connected at the top edge of interface boards by means of flat cable. These cables are routed over the top of the boards along the A and B columns and exit via a cutout at the top rear of the chassis. A strain-relief clamp is provided. All standard CDP-XI controller and memory boards have provision for this cable routing scheme.

2.4 SPECIFICATIONS

General specifications for the CDP-XI/00 are given in Table 2-2.

TM - QUADBOARD is a trademark of California Data Processors.

Table 2-2. CDP-XI/00 Processor Specifications

Characteristic	Specification
TYPE	High-speed microprogrammed digital computer designed for efficient emulation of general-purpose computer architectures and for direct custom applications.
CONTROL	
Microcommand length	48 bits.
Execution rate	150 ns min, 300 ns if skip or branch is made. (Clock rate is adjustable.)
Microcommand classes	16 arithmetic. 8 logical. 8 special.
Special operations	Special microcommands include double-precision shift, multiply-step and divide-step functions.
Conditional skip/branch	Each microcommand has conditional skip or branch capability. Tests can be made on current (dynamic) conditions or on previous (static) conditions.
Fixed control memory	Bipolar ROM or PROM; 4,096 words max.
Alterable control memory	Bipolar RAM; 512 words max without auxiliary power; 1,536 words max with auxiliary power.
Control memory stack	16-level hardware pushdown stack.
Emulation enhancement	Special emulation decode tables provide automatic addresses to control memory microroutines for high-speed program execution.
Loop counter	Eight-bit counter for single or multi-instruction repeats.
Interrupts	Multilevel priority-interrupt structure provides automatic addresses to control memory microroutines for internal and external conditions.
PROCESSING	
Word length	16 bits.
Arithmetic/logic	Both word and byte operations are provided; fixed point, one's or two's complement arithmetic; arithmetic

Table 2-2. (Continued)

Characteristic	Specification
	condition codes are carry (link), overflow, negative, zero, positive, odd; arithmetic and logical shifts (multibit using loop counter for repeats are provided).
Registers	Eight or sixteen 16-bit multipurpose files. Instruction register (IR). Decode register (ER). Processor (macro)status register (PS). Microstatus register (MS). MACROBUS address register/counter (AR). MACROBUS output data register/shifter (DR). MACROBUS input data-read register (RR).
INPUT/OUTPUT	
Type	Asynchronous bidirectional MACROBUS; master/slave interlocked control; handles all communications between processor, memory and peripheral elements.
Data	16 bits with byte capability.
Address	18 bits; least-significant bit for byte addressing.
MACROBUS priorities and requests	Nonprocessor request (NPR) for direct device-to-device transfers. Four full MACROBUS priority-request levels with multiple requests per level. Processor can set its own priority to any level except NPR.
Serial I/O channel (option)	Serial I/O controller for rates up to 4800 baud; RS-232 or current-loop interface.
Memory	Magnetic core; 8K or 16K words per module; 16 bits per word.
Memory expansion	124K words maximum with memory management unit (option).
Memory interleave	8K-word or 16K-word Cal Data core memory pairs can be interleaved for increased throughput rate.
Line-frequency clock	50/60-Hz line clock.

Table 2-2. (Continued)

Characteristic	Specification
PACKAGING	
Processor chassis	10-1/2 inches (26.7 cm) high by 19 inches (48 cm) wide by 24 inches (43 cm) deep; rack-mounted (slides) or table-top; vertical, top-loaded boards; contains macropanel, processor, plus additional slots for memory and I/O controllers; internal power supply; cooling fans; internal power distribution.
Connectors	36-pin, 0.6-inch (1.52-cm) card insertion depth; mounted on printed-circuit backplane.
Board size	8.94 inches by 15.7 inches (22.71 by 39.9 cm); six connector positions (216 pins) on long edge.
POWER	
AC input	115/208/230 Vac, 47 to 63 Hz.
DC outputs	Regulated: +5 Vdc, 36 A. -15 Vdc, 12 A. Unregulated: -22 Vdc, 1.5 A +8 Vrms, 1.5 A
Power monitor	Power-failure/restart signals to processor for automatic shutdown and restart operations.
ENVIRONMENT	0° to 50° C ambient temperature. 10 to 90% relative humidity, without condensation.
CIRCUITS	
Integrated circuits	Bipolar TTL; extensive MSI and LSI usage.
Discrete devices	Metal-can transistors; hermetically sealed components only.
Internal logic levels	ZERO = 0 Vdc; ONE = +5 Vdc, nominal.
I/O logic levels	ZERO = +3.4 Vdc, nominal; ONE = 0 Vdc.
MICROPROGRAMMING	
SUPPORT HARDWARE	
Micropanel	Provides direct control over processor; microcommand entry and display; single-step and trap-mode execution.

Table 2-2. (Continued)

Characteristic	Specification
Alterable control memory (ACM)	Modular 256-word increments of writable control memory; can be loaded and read; operates processor at full execution speed
PROM programmer	Modular plug-in board that permits PROM devices to be electrically programmed under processor control.
MICROPROGRAMMING SUPPORT SOFTWARE/ FIRMWARE	
Symbolic microassembler	Symbolic assembler for microprogram coding and documentation.
ACM software operating system	Operating system used in conjunction with ACM and PROM programmer.

SECTION 3
MACROBUS

3.1 GENERAL

The main interface channel of the CDP-XI system is the MACROBUS, used for parallel transfer of information and control signals among all functional system elements. An optional low-speed serial I/O data channel is also available. The significance of the MACROBUS in a system configuration is illustrated in Figure 3-1. This section describes only the MACROBUS.

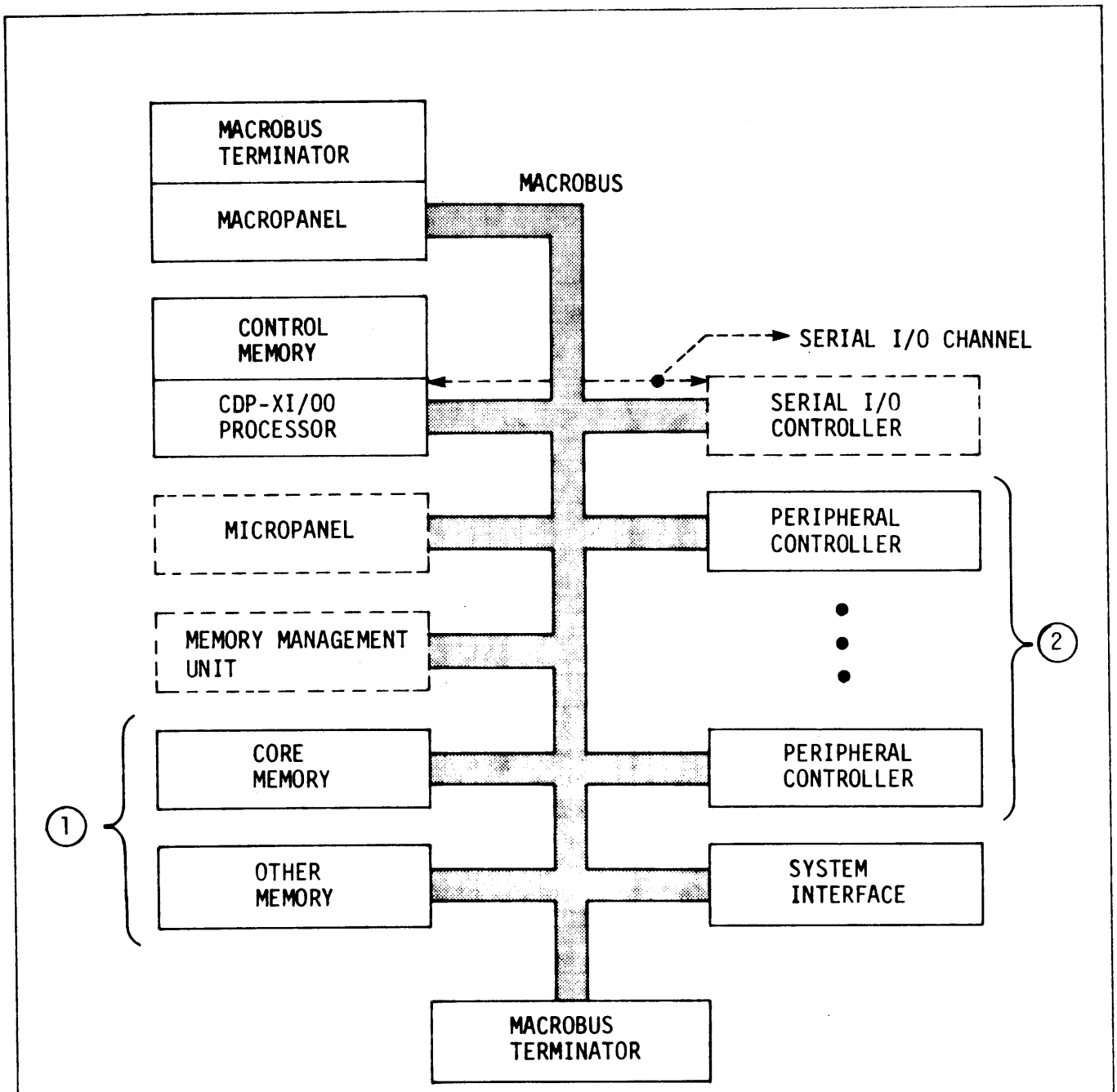
The MACROBUS is a universal, asynchronous, bidirectional I/O channel that can accommodate a wide variety of peripheral devices, memory units and other systems or processors. All components attached to the MACROBUS are treated as I/O devices by the CDP-XI/00 processor. MACROBUS devices can communicate directly with one another, as well as with the processor.

The bus consists of 56 lines that carry address, data and control information between system elements. All bus lines, except bus-grant lines, are bidirectional.

The MACROBUS is used by the processor and all I/O devices. Communication between two devices on the MACROBUS is in a master-slave relationship. During a bus operation, the master device has control of the MACROBUS when communicating with another device (the slave) on the bus. A priority structure determines which device obtains control of the MACROBUS, and every device on the MACROBUS capable of becoming bus master has an assigned priority. When two devices with identical priority levels simultaneously request use of the MACROBUS, the device that is electrically closest to the bus receives control.

Since all system elements are addressed in an identical manner on the MACROBUS, it is possible to implement direct operations between bus elements (controllers to/from memory, controllers to/from controllers) without involving the processor itself. Provision is made for priority transfers of bus mastership to devices requesting bus access. In addition, a priority-interrupt structure is provided to permit peripheral devices to interrupt the processor program.

Communication on the MACROBUS is interlocked, and each control signal issued by the master must be acknowledged by a response from the slave to complete the transfer. The MACROBUS transfer rate is thus automatically balanced among different devices. Communication is independent of the physical bus length and the response time of the master and slave devices. The maximum transfer rate on the MACROBUS, with minimum delay in each device, is one 16-bit word every 285 nanoseconds, or 3.5 million 16-bit words per second.



(1) Program addresses: 0 to 28K without memory management unit,
0 to 124K with memory management unit.
Cal Data core memory can be extended to 31K using the 7K or 15K memory option. (I/O device addresses are then 31K to 32K.)

(2) Program addresses: 28K to 32K.
Physical addresses: 124K to 128K.

Figure 3-1. MACROBUS Application

3.2 MACROBUS OPERATION

Because multiple I/O devices can be operating simultaneously, the MACROBUS is time-shared. Time sharing is controlled by the bus priority logic in the processor.

All requests for MACROBUS mastership are made to the processor, which performs the system "housekeeping". A device requesting MACROBUS access is given bus mastership when no higher-priority device is waiting. Once access is obtained, the bus master retains control of the MACROBUS until the necessary operation is completed. Control is then relinquished to the highest-priority device requesting access. Thus, the total available I/O bandwidth is automatically distributed among all devices sharing the MACROBUS. When no other device has control of the MACROBUS, mastership returns automatically to the processor.

A device other than the processor obtains bus mastership for one of two types of transfer:

a. To initiate a direct transfer between itself and another device without processor control. Nonprocessor transfers can interrupt the processor at the end of any bus cycle.
b. To interrupt the current processor program with a vector address that is the memory location of the interrupt routine required to service the device. This can take place only between non-priveleged instructions.

In a processor-assisted transfer, the subroutine vector address is transmitted to the processor, which also receives mastership of the MACROBUS. The processor then executes the subroutine. Thus, the device initiates the bus operation, but the actual data transfer is under processor control.

3.2.1 MACROBUS Priorities

When a device requests bus mastership, the handling of the request depends on the location of that device in the priority structure. The following factors determine the priority of the request:

a. The processor priority is set under program control to one of eight levels using bits 07, 06 and 05 in the PS register. The processor priority level inhibits recognition of bus requests on the same or lower levels.
b. MACROBUS requests from external devices can be made on any one of five request lines: Bus Request lines BR7 to BR4 and Non-processor Request line NPR. NPR has the highest priority and is granted by the processor between any two bus cycles of instruction execution (except after a DATIP operation). BR7 is the next-highest priority after NPR, and BR4 is the lowest. The four lower-level priority requests (BR7 to BR4) interrupt the processor only between instructions.
c. When multiple devices share the same bus request line, the device electrically closest to the processor has the highest priority. Any number of devices can be connected to a specific BR or NPR line, using a priority-chaining technique.

3.2.2 Processor Interrupts

External devices interrupt the processor via one of the Bus Request lines (BR7 to BR4). NPR cannot be used for a processor interrupt. The program sequence being performed by the processor is interrupted and an interrupt service routine is initiated. After the device request has been serviced, the program returns the processor to its former sequence.

The general operations performed in response to an interrupt request are:

1. The processor relinquishes MACROBUS control to the interrupting device.
2. When the device gains control of the MACROBUS, it sends the processor an interrupt request (INTR) and the vector address of a memory location that contains the starting address of the interrupt service routine. Immediately following this pointer address is a word (located at vector address +2) that becomes the new processor status.
3. The processor pushes the current processor status (PS) and the current program counter (PC) into a processor stack in memory.
4. New PC and PS values are taken from the vector locations specified by the interrupting device, and the interrupt service routine is initiated.
5. The interrupt sequence is performed in less than five microseconds from the time the interrupting device asserts INTR and places the vector address on the data lines until it initiates the fetch of the first instruction of the interrupt routine, assuming that no NPR transfers occur during this time.
6. The interrupt routine returns control to the interrupted sequence.
7. An interrupt routine can be interrupted in the same manner by a higher-priority interrupt, or can interrupt any time after the current PS and PC values have been pushed into the stack. Nesting of priority interrupts can continue to any level within the limits of memory space available for the processor stack.

3.2.3 MACROBUS Device Operation

Registers in MACROBUS devices are assigned bus addresses similar to memory locations (Appendix B). Thus, all CDP-XI program instructions that address memory locations are also I/O instructions. Control functions are assigned a register address, and the individual bits within that "register" govern the occurrence of control operations. Status conditions are also handled by the assignement of bits within a device register, and the status is checked with appropriate I/O instructions.

A peripheral device can request matership of the MACROBUS to:

a. Make a direct transfer of data or control information to or from another MACROBUS element.

b. Interrupt the processor program and force the processor to branch to the address of an interrupt service routine.

For nonprocessor transfer requests, the requesting and granting of next bus mastership is performed concurrently with current bus operations. While one device is using the MACROBUS, other nonprocessor requests are checked for priority. Program interrupt priorities are checked only on completion of a nonprivileged program instruction.

3.2.4 Direct Data Transfers

Direct-memory-access (DMA) data transfers can be accomplished between any two peripherals without processor intervention. A device can gain bus mastership for such a transfer using either an NPR or BR line, although an NPR is generally used. The processor program is not affected by this type of transfer.

In response to an NPR, the processor relinquishes MACROBUS control at the end of any bus cycle during execution of an instruction, except between cycles of a read-modify-write sequence (DATIP followed by DATO or DATOB). A bus master can make direct transfers of 16-bit words to or from memory at the memory cycle rate, or every 675 nanoseconds with the CDP-8KX16 core memory. With interleaved 8K memories, the times are faster: up to one word every 425 nanoseconds on transfers to memory, and every 340 nanoseconds on transfers from memory.

3.3 MACROBUS SIGNAL LINES

The MACROBUS consists of 54 active and two reserved signal lines. All devices, including the processor, are connected in parallel to these lines. On the bidirectional signal lines, signals flow in either direction. Five undirectional lines are used for priority bus control. The signal names and mnemonics are listed in Table 3-1.

3.3.1 Data-Transfer Lines

Thirty-eight bidirectional bus lines are used for data transfer. In a data transfer, the bus master controls the transfer of data to or from a slave device. The processor is bus master whenever no other device is using the MACROBUS, and it is master for all data transfers involved in normal instruction processing.

3.3.1.1 Data Lines

The 16 MACROBUS data lines used to transfer information between master and slave have the format:

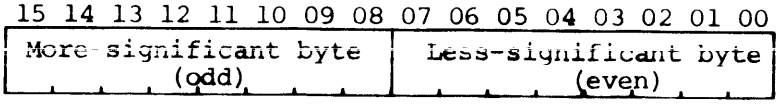


Table 3-1. MACROBUS Signals

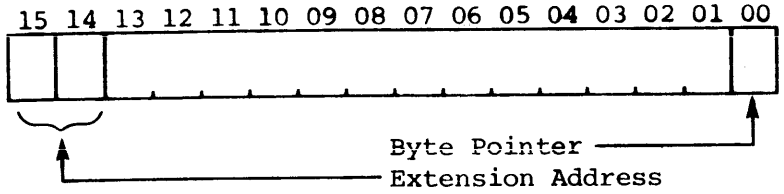
Type	Name	Mnemonic	Lines
Data Transfer	Data	D15 to D00	16
	Address	A17 to A00	18
	Mode Control	C1 and C0	2
	Master Synchronization	MSYN	1
	Slave Synchronization	SSYN	1
Priority Control	Nonprocessor Request	NPR	1
	Bus Request	BR7 to BR4	4
	Nonprocessor Grant	NPG	1
	Bus Grant	BG7 to BG4	4
	Selection Acknowledgement	SACK	1
	Bus Busy	BBSY	1
	Interrupt Request	INTR	1
Other	Initialization	INIT	1
	AC Low	ACLO	1
	DC Low	DCLO	1
	Reserved	PA	1
	Reserved	PB	1

Note:

Signals on the MACROBUS are asserted when low (except for the unidirectional bus-grant lines). All timing diagrams in this document reflect the asserted and cleared levels of MACROBUS lines.

3.3.1.2 Address Lines

Eighteen physical address lines are used by the master device to select a memory or device-register address. This permits expansion of the total bus address capacity to 131,072 words (262,144 bytes). The format of the 18 address lines is:



Lines A17 to A01 specify the address of a unique 16-bit word. In byte operations, A00 specifies the byte being referenced; A00 reset to ZERO references the less-significant (even) byte, and A00 set to ONE references the more-significant (odd) byte. Words are addressed only on even byte boundaries (i.e., A00 reset to ZERO).

Processor-generated addresses are a maximum of 16 bits in length, which permits direct access to only the first 32K-word (65K-byte) locations on the MACROBUS. Processor address bits A17 and A16 always contain ZEROs except for (word) addresses in the range 28K to 32K. In this range, these bits are automatically translated by the processor to generate physical MACROBUS (word) addresses in the range 124K to 128K. This upper range is reserved for nonmemory device addresses (Appendix B).

Peripheral controllers assuming bus mastership must directly generate a full 18-bit (if required) physical address.

3.3.1.3 Control Lines

The MACROBUS control signals are:

a. Mode Control lines. These two signals (C1 and C0) are coded by the master device to control the slave in one of four possible data transfer modes, as shown in Table 3-2.

b. Master and Slave Synchronization. Master Synchronization (MSYN) is a control signal used by the master device to indicate to the slave device that address and control information is present. Slave Synchronization (SSYN) is the slave response to the master (usually a response to MSYN).

Table 3-2. Data Transfer Modes

Name	Mnemonic	Control Signals C1	C0	Function
Data In	DATI	0	0	Data transfer from slave to master.
Data In, Pause	DATIP	0	1	Inhibits restore cycle in destructive readout devices. The pause flip-flop is set to inhibit the clear cycle on the following mandatory DATO or DATOB operation.
Data Out	DATO	1	0	Data transfer from master to slave.
Data Out, Byte	DATOB	1	1	Transfers data from the master to a single byte in the slave. Data is transmitted on D15 to D08 when A00 is ONE, or on D07 to D00 when A00 is ZERO.

3.3.2 Priority Control Lines

The MACROBUS contains 13 priority control lines. Four of these are the priority program interrupt Bus Request lines (BR7 to BR4) and one is the Nonprocessor Request line (NPR). The next five are the corresponding request grant lines (BG7 to BG4 and NPG), which the processor uses to respond to bus requests. Each device of the same priority level passes a grant signal to the next device on the line, unless the device has requested bus control in which case the requesting device blocks the grant signal from the following devices and assumes bus control.

There are three other control lines: SACK, BBSY and INTR (see below).

The priority control lines are:

a. Bus Request lines. These four signals (BR7 to BR4) are used by peripheral devices to request control of the MACROBUS in conjunction with the processor.

b. Bus Grant lines. These signals (BG7 to BR4) are the processor responses to MACROBUS requests. They are asserted only at the end of an instruction execution and in accordance with rules of priority.

c. Nonprocessor Request. This signal (NPR) is a bus request from a peripheral device used to gain MACROBUS control for data transfers not involving the processor.

d. Nonprocessor Grant. This signal (NPG) is the priority control response to an NPR. It can occur at the end of any MACROBUS cycle (other than DATIP).

e. Selection Acknowledgement. This signal (SACK) is asserted by a bus-requesting device that has received a bus grant. MACROBUS control passes to this device when the current bus master completes its operation. (If SACK is not received within 10 microseconds after a bus grant is issued, a time-out occurs and the bus grant is cleared automatically by the processor. This does not, however, cause a processor timeout error trap).

f. Interrupt Request. This signal (INTR) is asserted by the bus master to start a program interrupt operation in the processor. INTR can be asserted only by a device gaining control over the MACROBUS via a bus-request line, and not by an NPR.

g. Bus Busy. This signal (BBSY) is asserted by the bus master to indicate that the MACROBUS is being used.

3.3.3 Other Lines

Two additional lines on the MACROBUS that can be used by all devices are the Initialization (INIT) and AC Low (ACLO) lines. One additional signal available for devices that are mounted on the standard CDP-XI backplane and powered by the standard power supply is the DC Low (DCLO) line.

a. Initialization. This signal (INIT) is asserted by the processor when the START switch on the macropanel is pressed or when a power-failure sequence occurs. In the latter case, INIT is asserted following the power-failure service routine when power goes down and again when power comes up. INIT is also generated by the Reset microcommand.

b. AC Low. This signal (ACLO) starts the power-failure trap sequence and can also be used in peripheral devices to terminate operations prior to power loss. When ACLO is cleared, the power-up trap sequence in the processor begins. It is the programmer's responsibility to ensure that the trap vector location is loaded with the pointer to the power failure routine, otherwise an undefined sequence can result.

c. DC Low. This signal (DCLO) is generated by the processor power supply or by the processor power-failure/restart sequences. It remains cleared as long as all dc voltages are within specified limits. If an out-of-voltage condition occurs, DCLO is asserted by the power supply. Devices such as core memories use DCLO to inhibit further operations. DCLO is cleared before ACLO when power comes up and is asserted after ACLO when power goes down. The processor power-failure trap sequence is initiated by ACLO, while DCLO is used by the processor to assert INIT on the MACROBUS.

d. Reserved lines. These lines (PA and PB) are reserved by Cal Data for future use.

3.4 MACROBUS ACCESS

When any device, including the processor, requires MACROBUS access, it must first gain mastership of the MACROBUS. For devices other than the processor, this is done by generating a request signal on either the NPR or on one of the BR lines. Multiple requests for MACROBUS access are serviced on a priority basis as follows:

1. NPRs have the highest priority and are serviced between any two MACROBUS cycles, (except after DATIP), independent of the processor state. Priority of multiple requests on the single NPR line is determined by an electrical priority chain, where the requesting device closest to the MACROBUS is granted first access.
2. BR lines above the current processor priority level (specified in bits 07 to 05 of the PS register) have next-highest-priority. They are serviced only at the end of a processor instruction execution cycle.
3. The processor has priority over all requests on BR lines of the same or lower priority. Processor priority is made internal to the processor, which contains the priority control logic, hence no MACROBUS requests, as such, are made for processor access.

Priority transfers are basically of two types:

a. The requesting device can request a data transfer between itself and another MACROBUS device independently of the processor. This can be accomplished by making a bus request via either the NPR or one of the BR lines, depending on the priority assigned to the device. Once bus mastership is established and the MACROBUS is clear of any previous operation, the device performs the data transfer according to one of the sequences defined in subsection 3.5.

b. The requesting device can request a program interrupt to initiate a device service routine. This can be done only

via one of the BR lines. (A program interrupt sequence initiated via an NPR request causes improper system operation.) Once bus mastership is established, the MACROBUS is clear of any previous operation and the processor completes the current instruction, the new bus master (interrupting device) initiates an interrupt sequence.

3.4.1 Priority Data Transfers

Pricrity data transfers are initiated by a request on either NPR or a BR line. A signal sequence selects the device as next bus master when the device has the highest priority, and the appropriate data-transfer sequence follows.

Although BR lines can be used for priority transfers, the NPR is usually used for such operations. The MACROBUS access sequences illustrated in the following paragraphs are for NPR only. The same operations can occur using a BR line, the difference being recognition of the NPR between any two MACROBUS cycles versus recognition of a BR only after execution of a nonprivileged programmed instruction. The sequence of operations performed for selection of the next NPR bus master is defined below and in Figure 3-2:

1. The requesting device asynchronously asserts NPR.
2. Provided the SACK line on the MACROBUS is clear, the MACROBUS priority control logic immediately asserts NPG.
3. The first requesting device on the NPG line responds by asserting SACK, blocks NPG from all following devices and clears NPR.
4. When SACK is received, priority control logic clears NPG. (If SACK is not received within 10 microseconds, a timeout occurs and NPG is cleared automatically by the processor. This timeout does not result in a processor timeout error trap.) The requesting device, now selected as next bus master, continues to assert SACK until BBSY and MSYN are cleared by the current bus master on completion of its transfer, and SSYN is cleared by the current slave. Assertion of SACK by the selected device inhibits recognition of any other NPR by the priority control logic.
5. When BBSY, MSYN and SSYN are cleared after the previous MACROBUS transfer, the selected device asserts BBSY, clears SACK and initiates one of the data transfer sequences described in subsection 3.5.

The NPR sequence cannot be used to initiate a program interrupt, since indeterminate results occur.

An NPR device can make multiple data transfers after gaining control of the MACROBUS by leaving BBSY asserted and repeating the appropriate data transfer sequence. This occupies the MACROBUS during the entire interval.

3.4.2 Program Interrupts

A program interrupt can be generated by a device when it gains bus mastership via one of the BR lines. The device can preceded the

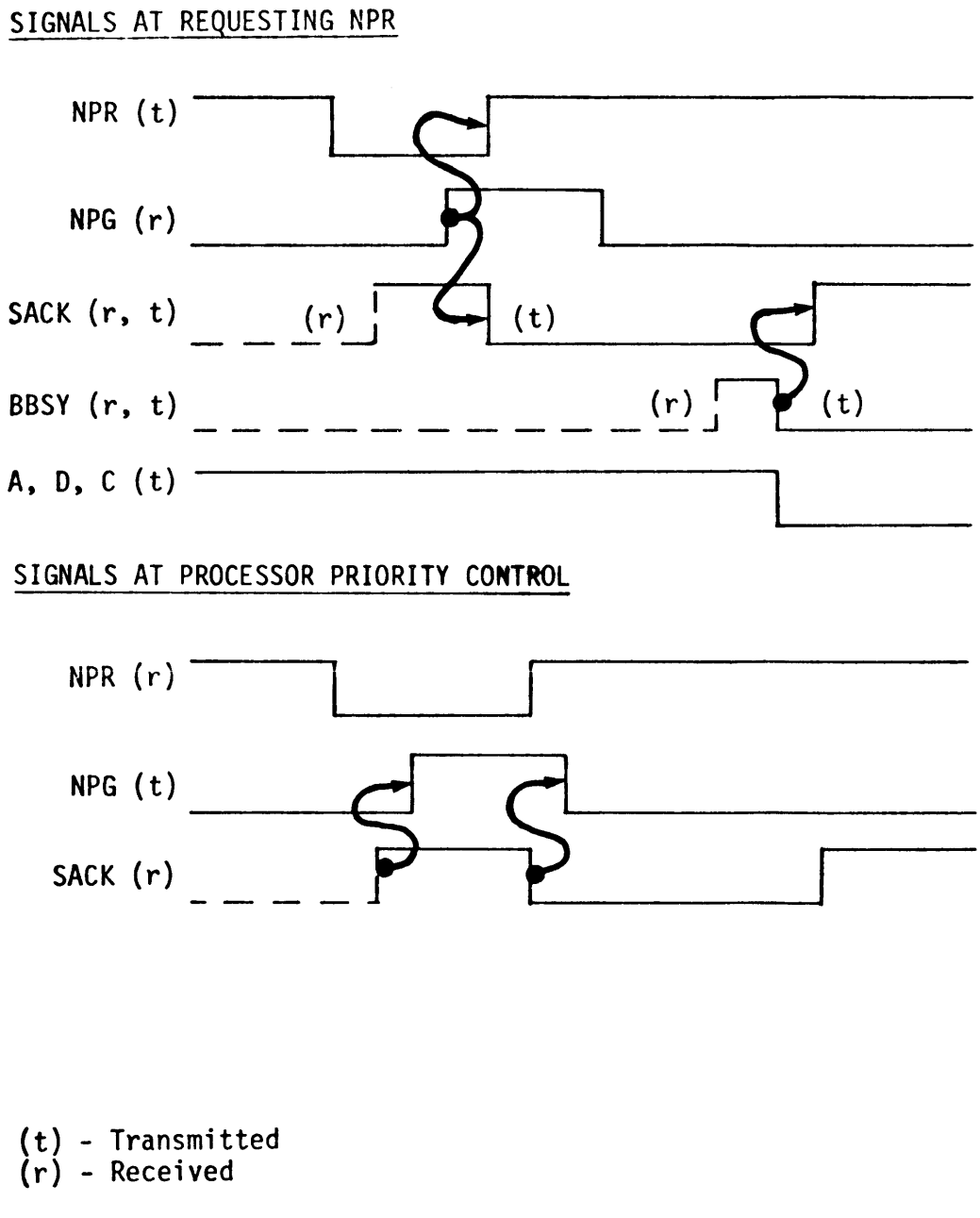


Figure 3-2. NPR Transfer Sequence

interrupt with one or more data transfers, if required. It must leave SACK asserted, however, until INTR is asserted, otherwise the request may not be recognized by the processor. The processor program is delayed during the period that SACK remains asserted by the device, and NPRs are not serviced during this period.

The sequence by which the device gains bus mastership is essentially the same as that previously described for an NPR. Figure 3-3 illustrates the sequence:

1. The requesting device asynchronously asserts its assigned BR.
2. BRs received by priority control are compared with the processor priority and with each other. At the end of an instruction execution cycle, and provided the SACK line is clear and no NPR is present, the priority control logic asserts a BG signal on the highest requested priority grant line above the current processor priority level.
3. The first requesting device on the asserted BG line responds by asserting SACK and blocks BG from all following devices. The device also clears BR at this time. (If SACK is not received within 10 microseconds, a timeout occurs and BG is cleared automatically by the processor. This does not result in a processor timeout error trap.)
4. When SACK is received, the priority control logic clears BG. The requesting device, now selected as the next bus master, continues to assert SACK until BBSY and MSYN are cleared by the current bus master on completion of its transfer, and SSYN is cleared by the slave.
5. When BBSY, MSYN and SSYN are cleared, the selected device asserts BBSY, then asserts INTR and places the interrupt vector address on the data lines at the same time that it clears SACK.
6. The processor receives INTR and asserts SSYN when it strobes the vector address. A 75-nanosecond delay is provided at the processor to ensure deskewing of the data lines.
7. The interrupting device clears INTR, BBSY and the data lines when SSYN is received from the processor, thus releasing the MACROBUS.
8. Assuming that an NPR has not gained bus mastership, the processor asserts BBSY on the MACROBUS and proceeds to execute access to the interrupt vector in memory, initiating the internal interrupt sequence.

3.5 DATA TRANSFER SEQUENCE

All MACROBUS transfers are asynchronous and depend on interlocking of control signals. In every case, a signal from a slave device is generated in response to a signal from a master device, and the master signal is then dropped in response to the slave signal. Signals on the MACROBUS are asserted when low (except for the unidirectional BG lines).

The four data-transfer modes are defined in Table 3-2. The bus master selects one of the four data transfers by asserting the proper code on the C1 and C0 lines. All data transfers are defined with reference to the master device (i.e., data-in is from slave to master and data-out is from master to slave).

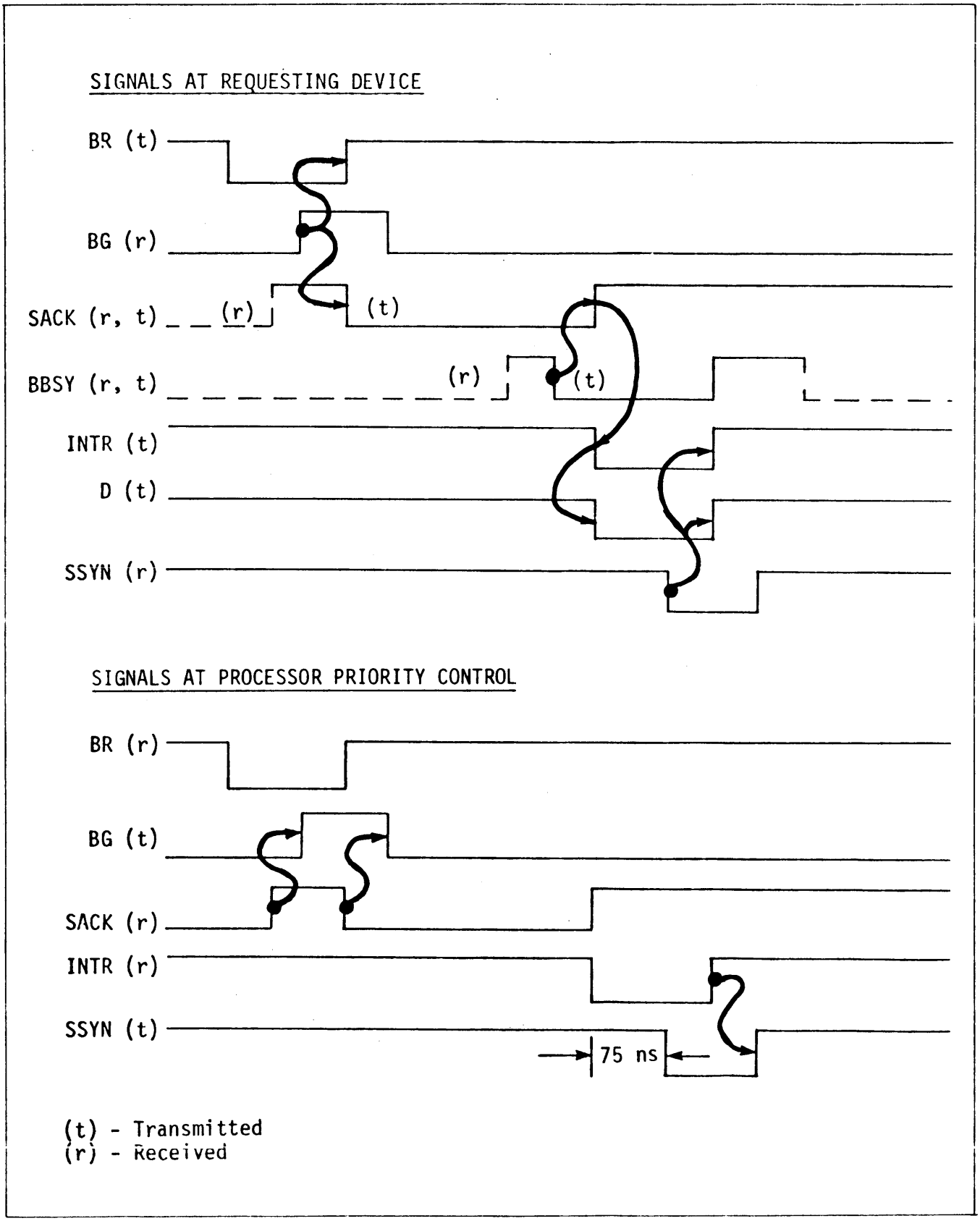


Figure 3-3. Interrupt Sequence

The master always delays MSYN to ensure that it does not reach the slave device prior to valid data or addresses. MSYN is further delayed 75 nanoseconds to allow for decoding by the slave device. Furthermore, to guarantee that there are no spurious selections, the master does not drop the address or control lines until 75 nanoseconds after MSYN has been dropped. When a slave transmits data to a master (DATI or DATIP), the deskew and decode time-delays are made by the master.

3.5.1 Data Input

Timing of the DATI and DATIP transfers after a device has gained bus mastership (subsection 3.4) is determined as follows and as shown in Figure 3-4:

1. If SSYN from a previous MACROBUS transfer is still asserted, the master waits until SSYN is clear. The master then places the address of the slave device on the address lines and the DATI or DATIP control code on C1 and C0. The master waits a minimum of 150 nanoseconds before asserting MSYN (75 nanoseconds for worst-case signal skew and 75 nanoseconds for slave-addressing decoding).
2. The slave decodes the address and control lines. When MSYN is received after address recognition, the slave accesses the specified data location for transfer of its contents to the master.
3. The slave places data, when available, on the data lines and asserts SSYN. In a DATI operation where the slave is a destructive read-out device (such as core memory), it immediately enters a restore cycle. For DATIP, the slave sets a pause flip-flop and waits for modified data before performing the restore cycle.

3.5.2 Data Output

Timing of the DATO and DATOB transfers after a device has gained bus mastership is determined as follows and as shown in Figure 3-5:

1. If SSYN from a previous MACROBUS transfer is still asserted, the master waits until SSYN is clear. The master then places the address (or location) of the slave device on the address lines, data on the data lines, and the DATO or DATOB control code on C1 and C0. The master waits a minimum of 150 nanoseconds before asserting MSYN (75 nanoseconds for worst-case signal skew and 75 nanoseconds for slave-address decoding).
2. The slave decodes the address and control lines. When MSYN is received after address recognition, the slave asserts SSYN when the transmitted data has been received. The slave accepts a full word for DATO and a byte for DATOB (the byte is specified by A00).
3. The master receives SSYN and clears MSYN. (If the processor is master and SSYN is not received within 20 microseconds, a timeout error trap occurs and MSYN is automatically cleared by the processor.)

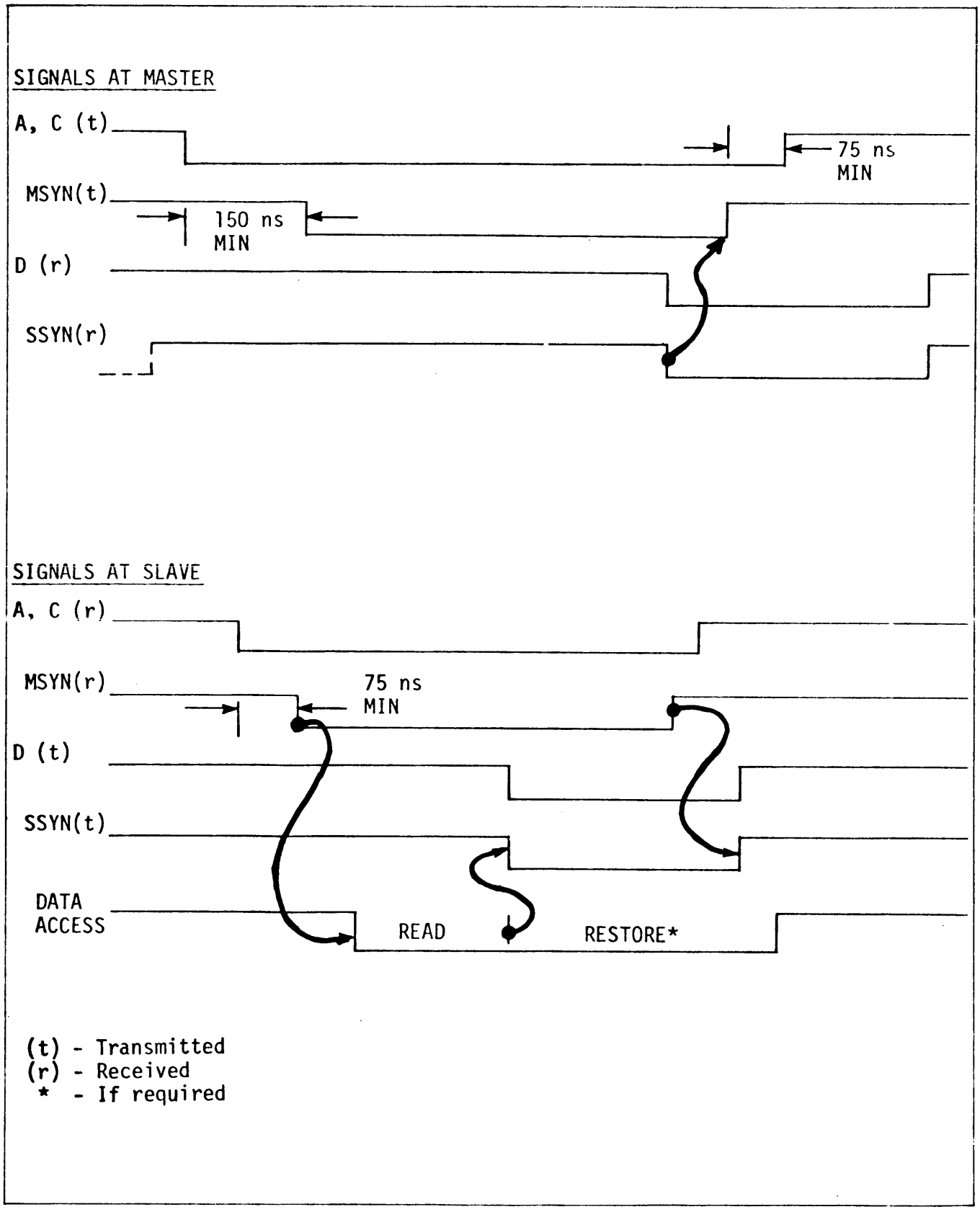


Figure 3-4. DATI, DATIP Timing Diagram

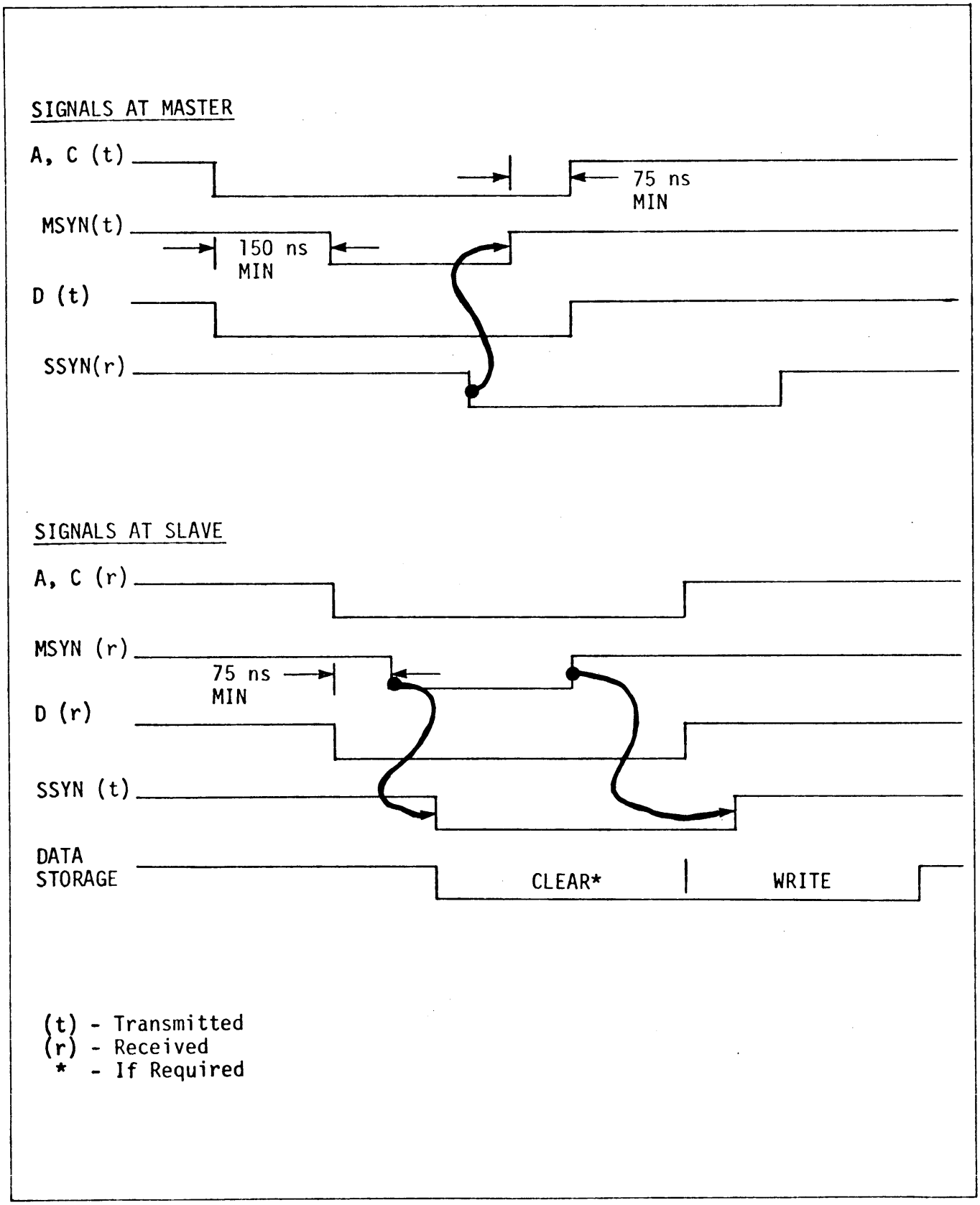


Figure 3-5. DATO, DATOB Timing Diagram

4. The master can modify the address, data and mode-control lines after a minimum delay of 75 nanoseconds from MSYN clear.
5. When MSYN is cleared, the slave clears SSYN.
6. The master receives cleared SSYN, completing the current MACROBUS transfer.

3.6 MACROBUS TIMING

The following paragraphs give the timing rules required for interfacing with the MACROBUS.

3.6.1 Standard Timing

To ensure total compatibility using different systems, the following standard design rules should be followed for master/slave interface design:

a. A 75-nanosecond delay must be provided to allow signals transferred in parallel (address and data) to settle prior to sampling the lines (deskewing time).
b. There must be a minimum of 75 nanoseconds from the time address and data lines completely settle at a slave device interface until MSYN is received. Thus, a master delays MSYN by at least 150 nanoseconds from the time it gates the address and data to the MACROBUS (75-nanosecond bus deskewing time, 75-nanosecond decoding time).
c. A slave can place data or an address on the MACROBUS at the same time that it responds with SSYN or INTR, respectively. The deskewing and decoding delays are handled by the master, except that the processor provides this delay for all of its transfers.
d. A master must delay clearing of address, data and control lines for 75 nanoseconds after SSYN is received from the slave.

3.6.2 Data Transfer Rates

The maximum transfer rate on the MACROBUS using these rules depends on the characteristics of the master and slave devices, and on the interface circuit designs.

The following examples illustrate minimum times for NPR data transfers by a master device having and maintaining control of the MACROBUS, to and from a slave device having essentially no read/access/write delay for data.

The data output rate (refer to Figure 3-4) is:

ACTION	NOMINAL TIME	COMMENT
Master asserts A, D and C lines.	-	
Master asserts MSYN.	150 ns	Standard minimum delay.
Slave strobes data, asserts SSYN.	30 ns	Assumes 7438 driver and Cal Data line receiver, typical delay, minimum-response design.

ACTION	NOMINAL TIME	COMMENT
Master clears MSYN.	30 ns	Assumes 7438 driver and Cal Data line receiver, typical delay, minimum-response design.
Master clears A, D and C.	75 ns	Standard delay.
	285 ns	

In the above example, a new transfer can start when the master clears the A, D and C lines, provided that SSYN has already been cleared by the slave. The maximum transfer rate to such a slave device is approximately 3.5 million words per second.

The data input rate (refer to Figure 3-5) is:

ACTION	NOMINAL TIME	COMMENT
Master asserts A and C lines.	-	
Master asserts MSYN.	150 ns	Standard minimum delay.
Slave returns data, asserts SSYN.	30 ns	Assumes 7438 driver and Cal Data line receiver, typical delay, minimum-response design.
Master deskews data, clears MSYN.	75 ns	Worst-case deskewing time assumed.
Master clears A and C.	75-ns	Standard minimum delay.
	330 ns	

In the above example, a new transfer can start when the master clears the A and C lines, provided that SSYN has already been cleared by the slave. The maximum transfer rate from such a slave device is approximately three million words per second.

Of more practical interest are the direct transfer rates associated with core memories. Using the Cal Data 8KX16 memory, it is possible to transfer words continuously at the cycle time of the memory. Using interleaved memory modules, the maximum rates are higher:

Single module:	675 ns	(1.48 million words per second)
Interleaved: DATO, DATOB:	425 ns	(2.35 million words per second)
DATI:	340 ns	(2.94 million words per second)

3.6.3 MACROBUS Timeout

Because failure to receive SSYN from a slave locks the MACROBUS indefinitely, every master should incorporate a timeout circuit. Delay time is dependent on the system requirements. Failure to receive SSYN within the timeout period should cause the master to clear all of its asserted signals and set an error flag in a status register.

SECTION 4
MACROPANEL

4.1 INTRODUCTION

The macropanel (Figure 4-1) contains the indicators and switches for operating the processor, a plastic cover imprinted with the designations of the indicators and switches, and the associated circuitry. This circuitry is mounted on a printed circuit board that plugs into the first slot of the processor chassis.

4.2 FUNCTIONAL DESCRIPTION

Functionally, the macropanel is a MACROBUS peripheral. It can thus be extended from the processor chassis. MACROBUS terminators on the macropanel board are provided for this configuration.

Interpretive microprogram routines become operational and service the macropanel when the system halts. These routines provide the basic system displays and controls, as well as additional convenience functions, and are completely transparent to the user.

The macropanel can operate the processor in run and manual modes. The step mode can be entered from only the micropanel, which is a microprogramming and maintenance aid described in a separate document.

The only active control switches in the program run mode are macropanel interrupt (INTR) and enable/halt (ENAB/HALT). In addition, the switch register can be read or tested by processor microcommands.

4.3 OPERATION

This section defines the use of the macropanel switches and indicators.

4.3.1 Power Switch (PWR ON/OFF)

The main ac line toggle switch on the rear of the power supply must be in the ON position before the macropanel power switch is effective.

The PWR switch applies and removes dc power from the processor. When the PWR switch is ON, all macropanel switches and indicators are functional. When PWR is set to OFF, there is no power to the processor.

The power supply has a lock slide switch accessible from the rear of the processor chassis. When this is in the lock position (down), power to the processor is on, but all macropanel switches (including PWR) are disabled, except the SWITCH REGISTER switches.

4.3.2 Address/Data Switch (ADDR/DATA)

ADDR/DATA determines whether MACROBUS addresses or data are displayed on the address/data indicators.

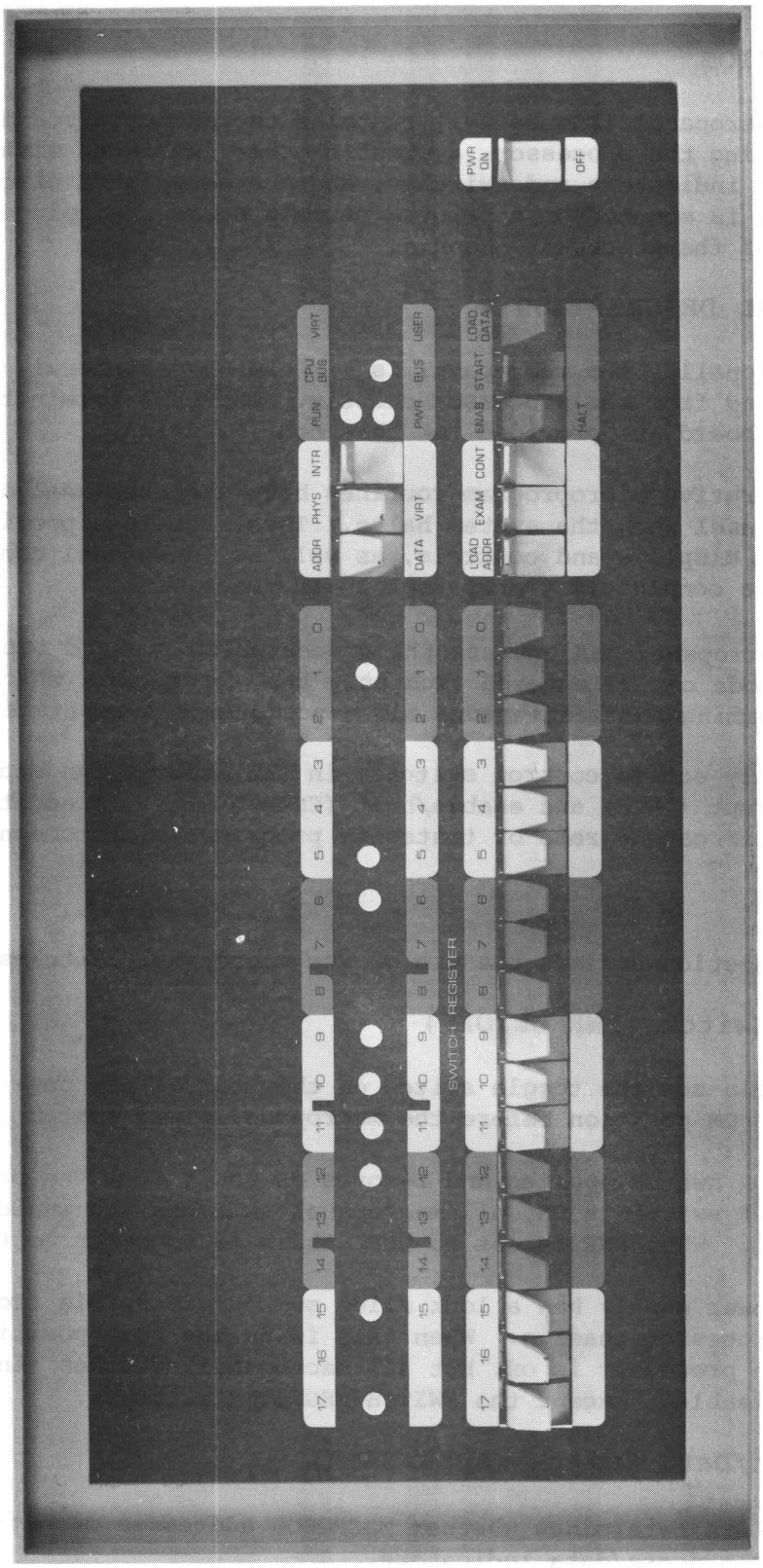


Figure 4-1. CDP-XI Macropanel

Setting ADDR/DATA up causes the currently selected MACROBUS address to be displayed. If an MMU is installed and enabled by the microprogram, the type of address displayed will be determined by the PHYS/VIRT switch.

Setting ADDR/DATA down causes the contents of the currently selected MACROBUS address to be displayed.

4.3.3 Physical/Virtual Address Switch (PHYS/VIRT)

PHYS/VIRT is activated only when the optional MMU is installed and enabled by the microprogram. PHYS/VIRT selects either a physical or virtual MACROBUS address for address examination or loading operations.

When PHYS/VIRT is up, the MMU address conversion is inhibited and the current macropanel address is used directly as the MACROBUS address.

When PHYS/VIRT is down and the MMU is enabled, the current macropanel address is treated as a virtual address that is relocated to a MACROBUS physical address by the MMU.

4.3.4 Address/Data Indicators

Eighteen light-emitting-diode (LED) indicators store and display either MACROBUS addresses or data, depending on the position of the ADDR/DATA switch.

When the program is running, the indicators display either the current bus address, with ADDR/DATA in the ADDR position, or the current bus data with ADDR/DATA in the DATA position.

When the program is halted manually from the macropanel, the indicators display either the next bus address or the current bus data, depending on the position of ADDR/DATA.

When the program is halted by a programmed microcommand, the indicators display either 0177540 (the macropanel switch-register address) or the current bus data, depending on the position of ADDR/DATA.

4.3.5 SWITCH REGISTER Switches

The 18 SWITCH REGISTER switches are used to manually place data and addresses on the MACROBUS. In the up position, each switch enters a ONE into the corresponding switch-register bit position. In the down position, a ZERO is entered.

The switch setting loaded manually can be displayed by using the ADDR/DATA switch; its position depending on what is being loaded.

When the optional MMU is installed in the system and enabled by the microprogram, SWITCH REGISTER address selection is affected by PHYS/VIRT. When PHYS/VIRT is up, the SWITCH REGISTER setting is stored in the processor as the starting MACROBUS address. When PHYS/VIRT is down, the SWITCH REGISTER setting is treated as a virtual address and stored in the MMU, where it is relocated to a MACROBUS physical address.

When the MMU is not enabled or not installed, the SWITCH REGISTER address selected is simply stored in the processor as the starting MACROBUS address.

4.3.6 Macropanel Interrupt Switch (INTR)

INTR interrupts the processor to vector address 0254 regardless of the processor priority level.

4.3.7 Load Address Switch (LOAD ADDR)

Pressing LOAD ADDR transfers the setting of SWITCH REGISTER to the processor MACROBUS address register (AR) or the MMU, depending on the status of the MMU and setting of PHYS/VIRT (see paragraph 4.3.5). The resulting MACROBUS address can be displayed by raising ADDR/DATA. This address is the starting address for operations using EXAM, START or LOAD DATA, and is not modified during program execution.

4.3.8 Examination Switch (EXAM)

Pressing EXAM displays the contents of the current MACROBUS address location if ADDR/DATA is down, or the address itself if ADDR/DATA is up.

If EXAM is pressed again, the contents of the next sequential address location (or the next address itself, depending on ADDR/DATA) are displayed. The MACROBUS address is automatically incremented by two each subsequent* time EXAM is pressed.

If an odd address is specified when EXAM is pressed and ADDR/DATA is down, the contents of the next lower even-address location are displayed.

4.3.9 Continuation Switch (CONT)

Pressing CONT restarts the processor, which continues operation from the halt point. If a programmed Halt instruction stopped the processor, CONT restarts the program without a system reset.

CONT is inoperative when the processor is in the run mode.

CONT steps the processor through single microcommands when the ENAB/HALT switch is in the HALT position (down).

4.3.10 Enable/Halt Switch (ENAB/HALT)

When ENAB/HALT is up, the processor performs normal operations under microprogram control (run mode).

Setting ENAB/HALT down (HALT) stops the processor. Pressing CONT then causes execution of a single microcommand (manual mode).

When ENAB/HALT is down, the START switch is inoperative, except for generating a bus Initialization signal each time it is pressed.

---

*The address is not incremented on the first use of EXAM after using any other command switch, such as LOAD DATA.

4.3.11 START Switch

If the program stops and ENAB/HALT is up, pressing START causes a system reset and restarts the program from the halt point.

The START switch is inoperative when the processor is in the run mode. When ENAB/HALT is set to HALT, pressing START generates an Initialization signal on the bus, but does not execute the next instruction.

4.3.12 LOAD DATA Switch

Raising LOAD DATA loads the contents of the switch register into the location specified by the MACROBUS address.

Raising LOAD DATA again loads the contents of the switch register into the next sequential address location. The MACROBUS address is automatically incremented (by two) each subsequent time LOAD DATA is raised. The contents of the switch register can be changed between uses of LOAD DATA.

If an odd address is specified when LOAD DATA is raised, the next lower even address is used as the specified location.

4.3.13 Status Indicators

The six status indicators (LEDs) are:

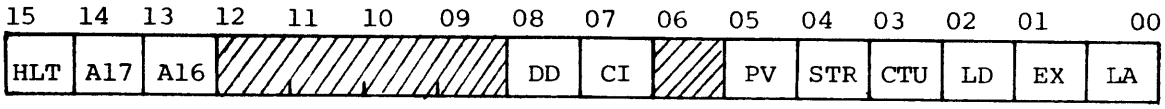
PWR	On when dc power is applied to the system and off otherwise.
RUN	On when the processor is executing a microprogram. RUN is off when the program is halted by an instruction or the HALT switch.
CPU BUS	On only when the processor has control of the MACROBUS and off otherwise.
BUS	On when any device, including the processor, has control of the MACROBUS.
VIRT	On when the PHYS/VIRT switch is in the VIRT position, and the MMU is installed and enabled (i.e., when the MMU is relocating virtual addresses).
USER	On when the MMU is installed and the system is operating in the User mode.

4.4 PROGRAMMING

The least-significant 16 bits of the switch register can be read as bus address 0177570.

Bus address 0177540 can be used to read the status of macropanel control switches plus bits 16 and 17 of the switch register. This location is also used to write (display) on the macropanel indicators the 16 bits of data or the address, according to the position of the ADDR/DATA switch.

The location of macropanel bits read via MACROBUS address 0177540 is:



BIT	FUNCTION	SWITCH TYPE
LA	Load address (LOAD ADDR)	Momentary
EX	Examine (EXAM)	Momentary
LD	Load Data (LOAD DATA)	Momentary
CTU	Continue (CONT)	Momentary
STR	Start (START)	Momentary
PV	Physical/virtual (PHYS/VIRT)	Two-position
CI	Macropanel interrupt (INTR)	Momentary
DD	Data display (ADDR/DATA)	Two-position
A16	Switch register bit 16	Two-position
A17	Switch register bit 17	Two-position
HLT	Enable/halt switch (ENAB/HALT)	Two-position

4.5 OPERATING THE MACROPANEL

The contents of MACROBUS address locations can be examined and modified only when the processor is halted. To examine a location, set SWITCH REGISTER to the address, ADDR/DATA to ADDR and press LOAD ADDR. The address is then displayed on the macropanel indicators. Set ADDR/DATA to DATA and press EXAM to display the contents of that address location.

To examine the contents of the next location in sequence, press EXAM again. The bus address is automatically incremented each *subsequent* time EXAM is pressed and always points to the address of the data currently being displayed.

To modify the data being displayed, set SWITCH REGISTER to the new bit configuration and raise LOAD DATA. Since this does not increment the bus address, the new data can now be displayed by simply pressing EXAM. Then, to examine the contents of the next location in sequence, press EXAM again and continue as before.

The address increments (by two) to the next MACROBUS word location in sequence when *successive* data examination or loading operations are performed. Interrupting a sequence of one type of operation to perform another prevents the address from incrementing until the new operation is repeated uninterruptedly.

To start a program, set SWITCH REGISTER to the starting address, set ADDR/DATA to ADDR and ENAB/HALT to ENAB, and press START.

To execute a single microcommand, set ENAB/HALT to HALT and press CONT. To execute successive microcommands one at a time, repeatedly press CONT. At this time, if ADDR/DATA is set to ADDR, the macropanel indicators display the current MACROBUS address. If ADDR/DATA is set to DATA, the indicators display the contents of the current bus address.

To return the processor to normal automatic running of the program, set ENAB/HALT to ENAB and press CONT.

If an attempt is made to gain macropanel access to a nonexistent bus location, a timeout trap to location 04 occurs and the data indicators display the contents of that location. To verify this, load a value other than that displayed from the trap location into the suspected bus location. If the contents of 04 are still displayed, there is either no such bus location or whatever device is assigned to it is malfunctioning.

To access the contents of file registers FR0 to FR7 or the PS register, use the bus addresses assigned to these registers (Appendix B). The bus addresses assigned to registers FR0 to FR7 apply to the macropanel only, and an attempt by the program to access these locations results in a timeout error trap.

SECTION 5
MICROCOMMANDS

5.1 GENERAL

Microcommands generate the control signals that enable all internal operations of the processor. There are no suboperations performed. All functions specified by a microcommand are executed simultaneously within a single clock step, with the following exceptions:

a. When the microprogram execution sequence is altered, one additional clock step is required to execute the branch operation.
b. A MACROBUS access delay inhibits microcommand execution until a synchronizing I/O response is received.

A clock period is 150 nanoseconds and all microcommands are executed within an integer multiple of that period.

The CDP-XI/00 incorporates a 48-bit microcommand word to perform all operations in the machine. The microcommand structure permits simulataneous execution of many parallel functions specified in each microcommand to achieve exceptionally fast emulation of general-purpose computer operations.

The structure of the microcommands provides considerable flexibility in organizing a particular microprogram to maintain high effective execution rates with economical use of control memory space.

5.2 MICROCOMMAND CLASSES

The three classes of microcommands are:

a. Logical.
b. Arithmetic.
c. Special.

Every microcommand, regardless of class, has the ability to specify a conditional or unconditional branch or skip operation. Since the format of the microcommands differs, depending on whether a branch or skip is specified, the microcommands in each class can be considered to be one of two types:

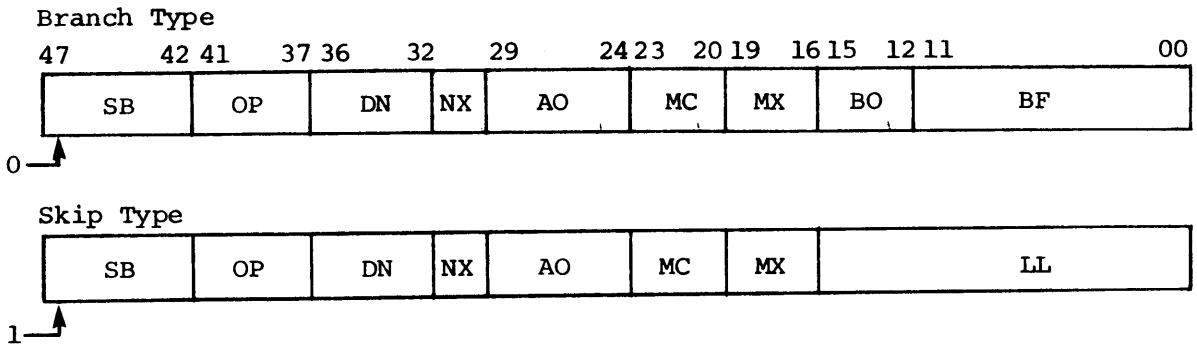
a. Branch type.
b. Skip type.

Figure 5-1 shows the formats for the classes and types of microcommands executed by the CDP-XI/00. The format for the logical and arithmetic classes is identical. The general characteristics of each class and type are defined below.

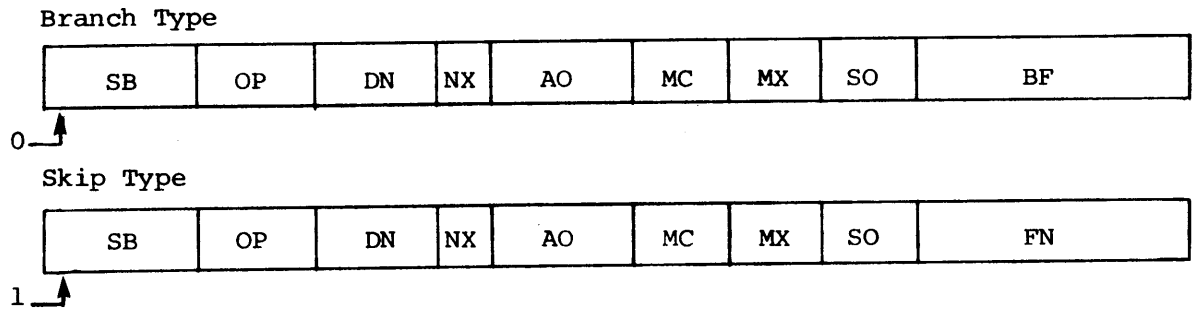
5.2.1 Logical and Arithmetic Classes

As the name implies, the logical and arithmetic classes of microcommands perform logical and arithmetic functions of one or two variables as specified by the microcommand. The specific logical or arithmetic

LOGICAL AND ARITHMETIC CLASSES



SPECIAL CLASS



SB = branch condition code (bit 47 specifies microcommand type).
OP = basic operation performed by the microcommand.
DN = destination address of result from the arithmetic/logic unit (AU).
NX = special control functions.
AO = source address of A operand to AU.
MC = microcondition code specification (dispositions).
MX = special control functions.
BO = source address of B operand to AU.
SO = special operation control functions.
BF = branch address or auxiliary control functions.
LL = literal value.
FN = auxiliary control functions.

Figure 5-1. Microcommand Formats

operation is defined by the OP field. A total of 16 logical and eight arithmetic operations are implemented. The same set of operations is performed regardless of whether a branch- or skip-type microcommand is used.

5.2.1.1 Branch Type

The logical or arithmetic branch-type microcommand permits the programmer to specify that a conditional or unconditional branch to a new program location can occur based on the results of executing the current microcommand (or on results previously stored).

In this type of microcommand, both an A and B operand to the AU are specified. The destination of the resulting operation is also specified. Arithmetic condition codes resulting from the microcommand execution can be saved or not.

If a branch condition is specified, an 11-bit branch address is provided that alters the program sequence if the branch condition is met. A control bit is also provided that can cause the next control memory address to be pushed into the control stack (CS) before the branch is made. This permits the microprogram to later execute an automatic return to the sequence via CS.

It is not necessary to specify a branch condition even though the microcommand is a branch type. If no branch condition is specified, an auxiliary set of control functions can be specified that are performed simultaneously with execution of the basic logical or arithmetic operation.

The remaining fields of the branch-type microcommand provide special control functions that can modify execution and content of the next microcommand in sequence. The operations performed by these fields are common to all microcommands, regardless of class and type.

5.2.1.2 Skip-Type

The logical or arithmetic skip-type microcommand performs the same basic operations as the branch type. The differences in the skip-type microcommands are:

a. Instead of a branch condition, a condition is specified under which execution of the microcommand at the next control memory location can be inhibited (skipped). The control memory address sequence itself is not altered.
b. The B-operand source and branch address are replaced by a 16-bit literal value. This value is used directly as the B operand for those logical and arithmetic operations that involve a B-operand input to AU.
c. Because of the space reserved for a literal value (whether or not one is required), the auxiliary control functions defined for the branch-type microcommand cannot be specified.

All other operations of a skip-type microcommand are identical to the corresponding branch-type microcommand.

5.2.2 Special Class

The special class of microcommands provides functions that affect specialized control and other operations required of the processor. Some of these microcommands involve the use of AU. The operation performed is specified by the OP field. A total of seven special operations are implemented.

5.2.2.1 Branch-Type

The special branch-type microcommand permits the programmer to specify a conditional or unconditional branch just as for the logical or arithmetic branch type. And, in the same way, either a branch address or a set of auxiliary control functions can be specified, depending on whether or not a branch condition is specified by the microcommand.

5.2.2.2 Skip-Type

The special skip-type microcommand is the same as the branch type and specifies the same operations, except that:

a. Instead of a branch condition, a condition is specified under which execution of the microcommand at the next control memory location can be inhibited (skipped). The control memory address sequence itself is not altered.

b. Since a branch address cannot be specified by this type of microcommand and since a B operand is never used, the space reserved for these is used to specify a set of auxiliary control functions.

5.3 MICROCOMMANDS

The following paragraphs present a description of each logical, arithmetic and special microcommand. A summary of all the basic microcommands executed by the CDP-XI/00 is given in Table 5-1.

The description of each microcommand includes the mnemonic, hexadecimal OP-field code, a symbolic notation describing its operation, where applicable, a description of the function performed, and examples or other comments to clarify the description.

The following symbols are used (in addition to many defined in Table 1-1):

| | = Absolute value of

( ) = contents of

$(_n)$ = contents of nth bit

$\overline{(\ )}$ = boolean complement

$\cap$ = boolean AND

$\cup$ = boolean OR

$\oplus$ = boolean exclusive OR

$<$ = less than

$=$ = equal to

$\neq$ = not equal to

+ = arithmetic addition (two's complement)

- = arithmetic subtraction (two's complement)

x = arithmetic multiplication

$\div$ = arithmetic division

$\geq$ = greater than or equal to

A = A operand to AU

B = B operand to AU

R = result (word)

R_M = result more-significant byte

R_L = result less-significant byte

CIN = carry input

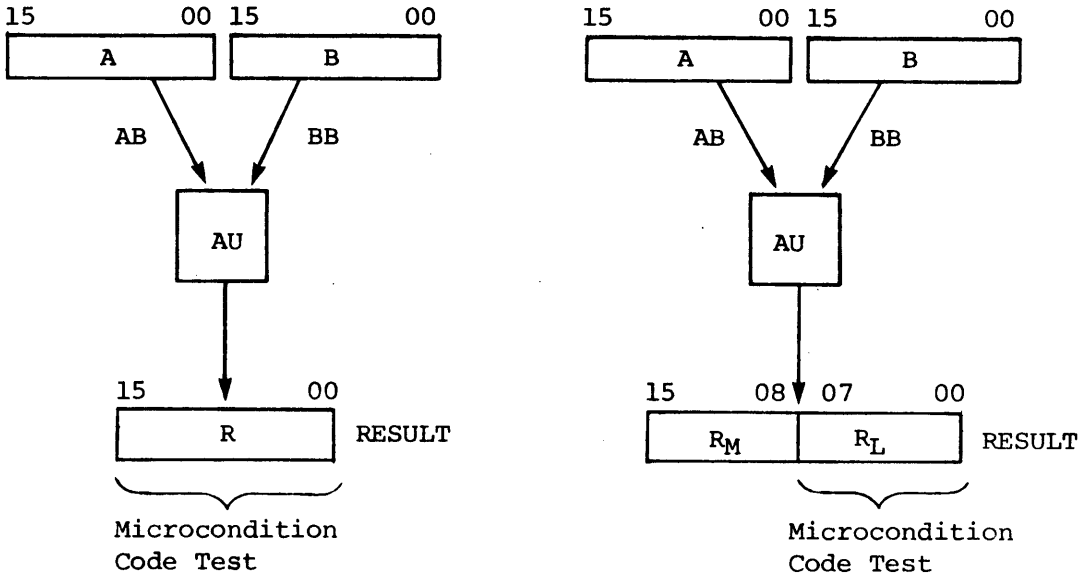
5.3.1 Logical Microcommands

The logical microcommands listed in Table 5-1 can be executed in either the word or byte mode. With one exception (SXA), the operation is performed on the full pair of operand words in AU and the 16-bit result is transferred to the destination via MB.

Table 5-1. CDP-XI/00 Microcommand Summary

Mnemonic	OP Field (Hexadecimal)	Name
		LOGICAL
EML	00	Emulate (optional)
SXA	01	Sign Extend A
MVA	02	Move A
MVB	03	Move B
OCA	04	Complement A
OCB	05	Complement B
AND	06	AND A, B
NDB	07	AND A, $\overline{B}$
NDA	08	AND $\overline{A}$, B
NOR	09	Not OR
ORI	0A	OR A, B
ORB	0B	OR A, $\overline{B}$
ORA	0C	OR $\overline{A}$, B
NAND	0D	Not AND
XOR	0E	Exclusive OR
COI	0F	Coincidence
		ARITHMETIC
ADD	10	Add, A, B
SUB	11	Subtract A, B
ADC	12	Add Carry
SBC	13	Subtract Carry
INC	14	Increase A
DEC	15	Decrease A
MSA	16	Add A Masked
TWB	17	Two's Complement B
		SPECIAL
SHF	18	Shift
MUS	19	Multiply Step
DVS	1A	Divide Step
TSB	1B	Test Bit
MMS	1C	Modify Macrostatus (optional)
CMA	1D	Conditional Memory Access (optional)
-	1E	(reserved)
DCD	1F	Decode

Microcondition codes are determined on the full word in the word mode and on the less-significant byte in the byte mode. This is illustrated below:



The microcondition codes for all logical-class microcommands are given in Table 5-2.

All logical microcommands are standard except Emulate (EML), which is optional. The main purpose of EML is for very rapid emulation decoding of instruction operation codes and control fields. The procedure is to store the command in IR. EML then initiates translation of the contents of IR into a control memory branch address generated from a table of values. The address directs the microprogram to the proper microroutine in control memory for the emulation of each instruction. The emulation table is specifically programmed for each computer to be emulated. The decoding operation performed by EML can be accomplished by other methods using only standard microcommands, but at the cost of time and control-memory space. For this reason, the need to implement EML depends on the specific application.

Table 5-2. Microcondition Codes for Logical Microcommands.

Microcondition Code	Definition	
	Word Mode	Byte Mode
c	always 0	always 0
v	always 1	always 1
z	1 if (R)=0; 0 otherwise	1 if (R_L)=0; 0 otherwise
n	1 if (R_{15})=1; 0 otherwise	1 if (R_{07})=1; 0 otherwise
p	1 if (R)>0; 0 otherwise	1 if (R_L)>0; 0 otherwise
d	1 if (R_{00})=1; 0 otherwise	1 if (R_{00})=1; 0 otherwise

5.3.1.1 Emulate (Optional)

Mnemonic: EML $00

Operation:
(R)=(A)
(R)→(DN)
(emulation table)→(cc), unless higher-priority CC modification occurs

Description: The A operand is transferred to the destination. The contents of IR are translated into a branch address directive to CC using an emulation table on the emulate board. If a higher-priority CC modification occurs concurrent with the microcommand, the emulation table address is ignored. All microcommand fields are effective as defined, except that the BO field is ignored, since no B operand is used.

5.3.1.2 Sign Extend A

Mnemonic: SXA $01

Operation:

Word mode	Byte mode
(R)=(A)	$(R_M)=(N) \cup (A_{15 \text{ to } 08})$
(R)→(DN)	$(R_L)=(A_{07 \text{ to } 00})$
	(R_M, R_L)→(DN)

Description: In the word mode, the A operand is transferred to the destination.

In the byte mode, the state of the negative microstatus bit, N, is extended to the more-significant byte of the A operand. The contents of the less-significant byte of the A operand are unmodified. The result is transferred to the destination.

Example: Perform a byte mode add on A and B and store the result in the A-operand location, then extend the sign of the byte result.

ADD:

```
    A   00000000 10110100  (-38)
   +B   00000000 11101100  (-10)
    R   00000001 10100000  (-48)
```

The microcondition codes generated are:
L = 1, V = 0, Z = 0, N = 1, P = 0, D = 0.

SXA:

```
    A   00000001 10100000
   UN   11111111
    R   11111111 10100000
        ---------- --------
            R        R_L
```


5.3.1.3 Move A

Mnemonic: MVA $02

Operation: (R)=(A)
(R)→(DN)

Description: The A operand is transferred unmodified to the destination.

5.3.1.4 Move B

Mnemonic: MVB $03

Operation: (R)=(B)
(R)→(DN)

Description: The B operand is transferred unmodified to the destination.

5.3.1.5 Complement A

Mnemonic: OCA $04

Operation: $(R)=(\overline{A})$
(R)→(DN)

Description: The logical or one's complement of the A operand is transferred to the destination.

Example:

	Binary	Octal	Hexadecimal
A	0110110100101100	0066454	$6D2C
R	1001001011010011	0111323	$92D3

5.3.1.6 Complement B

Mnemonic: OCB $05

Operation: $(R)=(\overline{B})$
(R)→(DN)

Description: The logical or one's complement of the B operand is transferred to the destination.

5.3.1.7 AND A, B

Mnemonic: AND $06

Operation: $(R)=(A)\cap(B)$
$(R)\rightarrow(DN)$

Description: The logical AND of the A and B operands is transferred to the destination.

5.3.1.8 AND A, $\overline{B}$

Mnemonic: NDB $07

Operation: $(R)=(A)\cap(\overline{B})$
$(\overline{R})\rightarrow(DN)$

Description: The logical complement of the B operand is ANDed with the A operand and the result is transferred to the destination.

5.3.1.9 AND $\overline{A}$, B

Mnemonic: NDA $08

Operation: $(R)=(\overline{A})\cap(B)$
$(R)\rightarrow(DN)$

Description: The logical complement of the A operand is ANDed with the B operand and the result is transferred to the destination.

5.3.1.10 Not OR

Mnemonic: NOR $09

Operation: $(R)=\overline{(A)\cup(B)}$
$(R)\rightarrow(DN)$

Description: The logical NOR of the A and B operands is transferred to the destination.

5.3.1.11 OR A, B

Mnemonic: ORI $0A

Operation: $(R)=(A)\cup(B)$
$(R)\rightarrow(DN)$

Description: The logical OR of the A and B operands is transferred to the destination.

5.3.1.12 OR A, $\overline{B}$

Mnemonic: ORB $0B

Operation: $(R)=(A)\cup(\overline{B})$
$(R)\rightarrow(DN)$

Description: The logical complement of the B operand is ORed with the A operand and the result is transferred to the destination.

5.3.1.13 OR $\overline{A}$, B

Mnemonic: ORA $0C

Operation: $(R)=(A)\cup(B)$
$(R)\rightarrow(DN)$

Description: The logical complement of the A operand is ORed with the B operand and the result is transferred to the destination.

5.3.1.14 Not AND

Mnemonic: NAND $0D

Operation: $(R)=\overline{(A)\cap(B)}$
$(R)\rightarrow(DN)$

Description: The logical NAND of the A and B operands is transferred to the destination.

5.3.1.15 Exclusive OR

Mnemonic: XOR $0E

Operation: $(R)=(A)\oplus(B)$
$(R)\rightarrow(DN)$

Description: The logical exclusive OR of the A and B operands is transferred to the destination. The exclusive OR by definition is:
$$(A)\oplus(B)=[(A)\cap(B)]\cup[(\overline{A})\cap(\overline{B})]$$

5.3.1.16 Coincidence

Mnemonic: COI $0F

Operation: $(R)=\overline{(A)\oplus(B)}$
$(R)\rightarrow(DN)$

Description: The complement of the logical exclusive OR of the A and B operands is transferred to the destination. This is the coincidence function:
$$\overline{(A)\oplus(B)}=[(\overline{A})\cap(B)]\cup[(A)\cap(\overline{B})]$$

5.3.2 Arithmetic Microcommands

The arithmetic microcommands are listed in Table 5-1. There are eight microcommands in this class.

The CDP-XI/00 performs both binary addition and subtraction (as opposed to complementary addition). Negative numbers are assumed to be represented as two's complements of positive numbers (although one's complement arithmetic can be performed, since the programmer has independent control of the carry and borrow inputs to AU). A complete description of binary arithmetic operations in the CDP-XI/00 is given in Appendix A. The carry and overflow microcondition codes differ for the addition and subtraction operations, as does the use of the carry-in term. The data value microcondition codes (z, n, p and d) are the same for addition and subtraction and depend only on the value of the arithmetic result.

Arithmetic operations can be executed in either the word or byte mode. In either mode, the specified operation is performed on the full pair of operand words in AU. The 16-bit result is transferred to the destination via MB. The microcondition codes are determined on the full word in the word mode and on the less-significant byte in the byte mode (see illustration in paragraph 5.3.1). The microcondition codes for addition and subtraction operations are defined in Table 5-3.

Table 5-3. Microcondition Codes for Arithmetic Microcommands

Micro-Condition Code	Arithmetic Operation	Definition	
		Word Mode	Byte Mode
c	Addition	$[(A_{15}) \cap (\overline{R_{15}})] \cup [(B_{15}) \cap (\overline{R_{15}})] \cup [(A_{15}) \cap (B_{15})]$	$[(A_{07}) \cap (\overline{R_{07}})] \cup [(B_{07}) \cap (\overline{R_{07}})] \cup [(A_{07}) \cap (B_{07})]$
	Subtraction	$[(\overline{A_{15}}) \cap (R_{15})] \cup [(\overline{A_{15}}) \cap (B_{15})] \cup [(B_{15}) \cap (R_{15})]$	$[(\overline{A_{07}}) \cap (R_{07})] \cup [(\overline{A_{07}}) \cap (B_{07})] \cup [(B_{07}) \cap (R_{07})]$
v	Addition	$[(A_{15}) \cap (B_{15}) \cap (\overline{R_{15}})] \cup [(\overline{A_{15}}) \cap (\overline{B_{15}}) \cap (R_{15})]$	$[(A_{07}) \cap (B_{07}) \cap (\overline{R_{07}})] \cup [(\overline{A_{07}}) \cap (\overline{B_{07}}) \cap (R_{07})]$
	Subtraction	$[(\overline{A_{15}}) \cap (B_{15}) \cap (R_{15})] \cup [(A_{15}) \cap (\overline{B_{15}}) \cap (\overline{R_{15}})]$	$[(\overline{A_{07}}) \cap (B_{07}) \cap (R_{07})] \cup [(A_{07}) \cap (\overline{B_{07}}) \cap (\overline{R_{07}})]$
z	Addition or Subtraction	1 if (R)=0; 0 otherwise.	1 if (R_L)=0; 0 otherwise.
n	Addition or Subtraction	1 if (R_{15})=1; 0 otherwise.	1 if (R_{07})=1; 0 otherwise.
p	Addition or Subtraction	1 if (R)>0; 0 otherwise.	1 if (R_L)>0; 0 otherwise.
d	Addition or Subtraction	1 if (R_{00})=1; 0 otherwise.	1 if (R_{00})=1; 0 otherwise.

5.3.2.1 Add A, B

Mnemonic: ADD $10

Operation:
(R)=(A)+(B)+CIN
(R)→(DN)

Description: The A and B operands and the value of CIN designated by the MC field are added arithmetically and the result is transferred to the destination.

Micro-condition Codes: Addition (Table 5-3).

Example: Add A and B and increment the result:

```
    A = +27,435 =   0110101100101011
   +B = - 1,747 =   1111100100101101
 +CIN = +     1 =   0000000000000001
    R = +25,689 = 1 0110010001011001
                  |
                  └──►c
```

The microcondition codes generated are:
c = 1, v = 0, z = 0, n = 0, p = 1, d = 1.

5.3.2.2 Subtract A, B

Mnemonic: SUB $11

Operation:
(R)=(A)-(B)-CIN
(R)→(DN)

Description: The B operand and the value of CIN designated by the MC field are subtracted from the A operand and the result is transferred to the destination.

Micro-condition Codes: Subtraction (Table 5-3).

Example: Subtract B from A and decrement the result:

```
    A =      -444 =     1111111001000100
   -B = -(-1,747) =   - 1111100100101101
 -CIN =     -(+1) =   - 0000000000000001
    R =    +1,302 =   0 0000010100010110
                      |
                      └──►c
```

This operation produces a one's complement result when the result is negative.

The microcondition codes generated are:
c = 0, v = 0, z = 0, n = 0, p = 1, d = 0.

5.3.2.3 Add Carry

Mnemonic: ADC $12

Operation: (R)=(A)+CIN
(R)→(DN)

Description: The value of CIN designated by the MC field is added to the A operand and the result is transferred to the destination.

Micro-condition Codes: Addition (Table 5-3).

5.3.2.4 Subtract Carry

Mnemonic: SBC $13

Operation: (R)=(A)-CIN
(R)→(DN)

Destination: The value of CIN designated by the MC field is subtracted from the A operand and the result is transferred to the destination.

Micro-condition Codes: Subtraction (Table 5-3).

5.3.2.5 Increase A

Mnemonic: INC $14

Operation: (R)=(A)+1+CIN
(R)→(DN)

Description: The value one and the value of CIN designated by the MC field are added to the A operand and the result is transferred to the destination. If CIN is ONE, the A operand is increased by two; otherwise, it is increased by one.

Micro-condition Codes: Addition (Table 5-3).

Examples: Increase the A operand by two if MC designates CIN as ONE; increase by one otherwise:

```
    A = +7817 =  0001111010001001
   +1 =    +1 =  0000000000000001
 +CIN =     0 =  0000000000000000
    R = +7818 =  0001111010001010
```

The microcondition codes generated are:
c = 0, v = 0, z = 0, n = 0, p = 1, d = 0.

Another example, where overflow is affected:

```
    A = +32,766 = 0111111111111110
   +1 = +     1 = 0000000000000001
 +CIN = +     1 = 0000000000000001
    R = +32,768 = 1000000000000000
```

The microcondition codes generated are:
c = 0, v = 1, z = 0, n = 1, p = 0, d = 0.

5.3.2.6 Decrease A

Mnemonic: DEC $15

Operation:
(R)=(A)-1-CIN
(R)→(DN)

Description: The quantity one and the value of CIN designated by the MC field are subtracted from the A operand and the result is transferred to the destination. If CIN is ONE, the A operand is decreased by two; otherwise, it is decreased by one.

Micro-condition Codes: Subtraction (Table 5-3).

Example: Decrease the A operand by two if the link microstatus bit is set; by one otherwise:

a. If (L) = 1:

```
   A =    +1 =   0000000000000001
  -1 = -(+1) = - 0000000000000001
 CIN = -(+1) = - 0000000000000001
   R = -(+1) = 1 1111111111111111
                 └──►c
```

The microcondition codes generated are:
c = 1 (borrow), v = 0, z = 0, n = 1,
p = 0, d = 1.

b. If (L) = 0:

```
   A =    +1 =   0000000000000001
  -1 = -(+1) = - 0000000000000001
 CIN =    -0 = - 0000000000000000
   R =     0 =   0000000000000000
```

The microcondition codes generated are:
c = 0, v = 0, z = 1, n = 0, p = 0, d = 0.

5.3.2.7 Add A Masked

Mnemonic: MSA $16

Operation: (R)=(A)+[(A)∩(B)]+ CIN
(R)→(DN)

Description: The logical AND of the A and B operands is added to the A operand and to the value of CIN designated by the MC field, and the result is transferred to the destination.

Micro-condition Codes: Addition (Table 5-3).

Example: Add the absolute value of the less-significant byte of A to the A operand:

```
    A =   -110 = 1111111110010010
    B =   Mask = 0000000001111111
  A∩B =    +18 = 0000000000010010
   +A =+(-110) = 1111111110010010
 +CIN =     +0 = 0000000000000000
    R =    -92 = 1111111110100100
```

The microcondition codes generated are:
c = 0, v = 0, z = 0, n = 1, p = 0, d = 0.

5.3.2.8 Two's Complement B

Mnemonic: TWB $17

Operation: (R)=-(B)-CIN
(R)→(DN)

Description: The value of CIN designated by the MC field is subtracted from the two's complement of the B operand and the result is transferred to the destination. If the value of CIN is ZERO, the result is the two's complement; otherwise, the result is the one's complement.

Micro-condition Codes: Subtraction (Table 5-3).

Example: Form the two's complement of B=+3131 if the link microstatus bit is ZERO. If the link bit is ONE, form the one's complement.

a. If (L)=0:

```
   -B = -3131 =   1111001111000101
 -CIN = -   0 = - 0000000000000000
    R = -3131 =   1111001111000101
```

The microcondition codes generated are:
c = 0, v = 0, z = 0, n = 1, p = 0, d = 1.

b. If (L)=1:

```
   -B = -3131 =   1111001111000101
  CIN = -(+1) = - 0000000000000001
    R = -3132 =   1111001111000100
```

The microcondition codes generated are:
c = 0, v = 0, z = 0, n = 1, p = 0, d = 0.

5.3.3 Special Microcommands

The seven special microcommands listed in Table 5-1 provide a powerful extension of the basic logical and arithmetic microcommands. Four of these are standard and have general application in all emulation microprograms. Three microcommands are defined as optional, since they must be tailored to a particular emulation system. The hardware elements that implement the optional microcommands are modularized to permit them to be either omitted or redefined without affecting the basic hardware of the system.

Microcondition codes generated for the special microcommands are generally identical to those defined for the arithmetic microcommands. The overall uses and limitations of the special microcommands are described in the following paragraphs.

5.3.3.1 Shift

The Shift microcommand provides complete flexibility for single-length shifts involving only the A operand and AU, and double-length shifts involving AU and DR. Shifts can be left or right, logical or arithmetic, open or closed. While the basic microcommand shifts only a single bit, multibit shifts can be performed by repeating the microcommand using LC.

Mnemonic: SHF $18

Microcommand Type: Special branch or special skip.

Description: The SO field specifies the type and direction of shift as shown in Table 5-4. For a single-precision (16-bit) shift, the A operand is shifted one place left or right and the result is transferred to the destination. For a double-precision (32-bit) shift, the A operand and the contents of DR are shifted one place left or right with a linked carry between the two words. The shifted AU result is transferred to the destination and the shifted DR result remains in DR. The shift operation is performed in the word mode unless a byte operation is specified by the FN field.

Using the special skip-type format for the shift, can lead to possible conflict between a double-length shift specification in the SO field and a DR shift specification in the FN field. If such a conflicting specification is made, the SO field control is effective and the FN field control is ignored.

Table 5-4. SO-Field Shift Specification

SO Field Bits 15	14	13	12	Shift Operation
0	0	x	x	swap halves
0	1	x	x	shift left, logical
1	0	x	x	shift right, logical
1	1	x	x	shift right, arithmetic
x	x	0	x	single precision
x	x	1	x	double precision
x	x	x	0	open shift
x	x	x	1	closed shift

5.3.3.1.1 Single-Precision Shifts. Single-precision shifts involve only AU shift elements operating on the A operand.

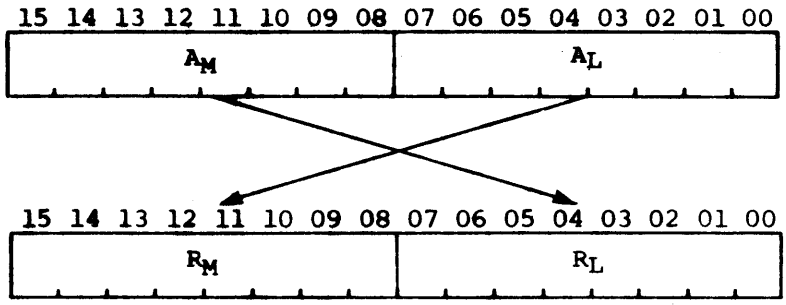
Swap Halves (Word or Byte).

Operation: a. Word mode:

$(R_{15\ to\ 08}) = (A_{07\ to\ 00})$

$(R_{07\ to\ 00}) = (A_{15\ to\ 08})$

$(R) \rightarrow (DN)$



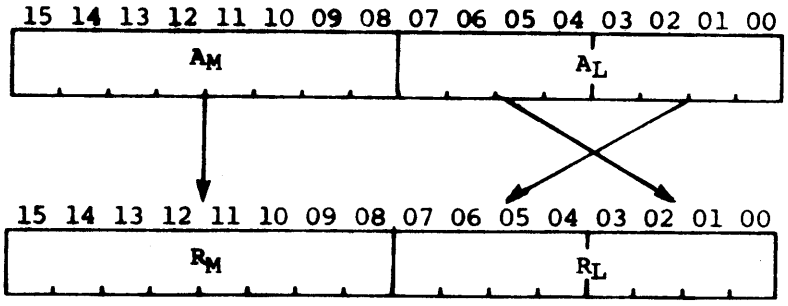
b. Byte mode:

$(R_{15\ to\ 08}) = (A_{15\ to\ 08})$

$(R_{07\ to\ 04}) = (A_{03\ to\ 00})$

$(R_{03\ to\ 00}) = (A_{07\ to\ 04})$

$(R) \rightarrow (DN)$



Description: For word swaps, the more-significant and less-significant bytes of the A operand are swapped and the result is transferred to the destination.

For byte swaps, the more-significant and less-significant halves of the less-significant byte of the A operand are swapped and the result is transferred to the less-significant byte of the desitnation. The more-significant byte of the A operand is transferred, unchanged, to the more-significant byte of the destination.

Micro-condition Codes:

a. Word mode:

c = (A_{15})
v = $(A_{15}) \oplus (A_{14})$
z = 1 if (R)=0;
0 otherwise
n = 1 if (R_{15})=1;
0 otherwise
p = 1 if (R)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

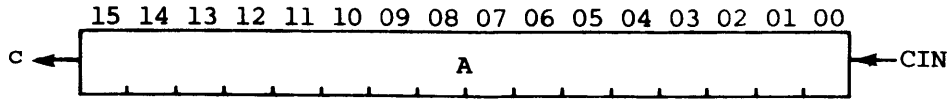
b. Byte Mode:

c = (A_{07})
v = $(A_{07}) \oplus (A_{06})$
z = 1 if (R_L)=0;
0 otherwise
n = 1 if (R_{07})=1;
0 otherwise
p = 1 if (R_L)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

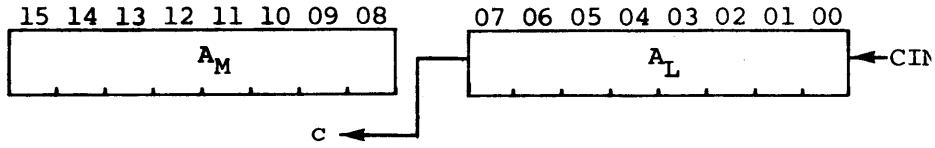
Logical Open Left Shift (Word or Byte).

Operation:

a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand is shifted left one bit. The value of CIN designated by the MC field is shifted into bit 00. The bit shifted out of bit 15 is the shift carry-out. The result is transferred to the destination.

Byte shifts are the same as word shifts, except that the shift is on the less-significant byte only. The carry bit is shifted out of bit 07. The more-significant byte is unmodified. The resulting word is transferred to the destination.

Micro-condition Codes:

a. Word Mode:

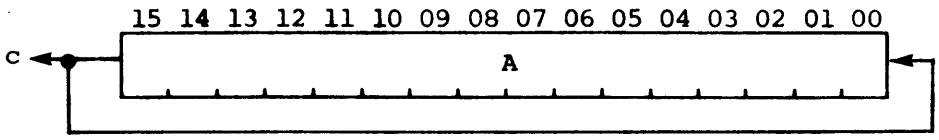
c = (A_{15})
v = $(A_{15}) \oplus (A_{14})$
z = 1 if (R)=0;
0 otherwise
n = 1 if (R_{15})=1;
0 otherwise
p = 1 if (R)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

b. Byte mode:

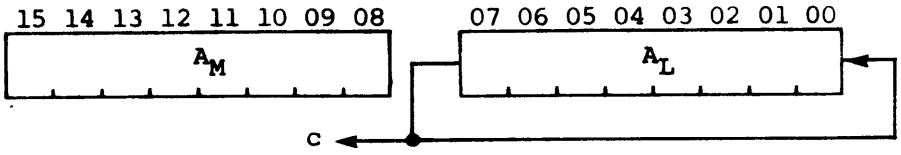
c = (A_{07})
v = $(A_{07}) \oplus (A_{06})$
z = 1 if (R_L)=0;
0 otherwise
n = 1 if (R_{07})=1;
0 otherwise
p = 1 if (R_L)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

Logical Closed Left Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand is shifted left one bit. Bit 15 is the shift carry-out and is also shifted into bit 00. The result is transferred to the destination.

Byte shifts are the same as word shifts, except that the shift is on the less-significant byte only. The carry bit is shifted out of bit 07. The more-significant byte is unmodified. The resulting word is transferred to the destination.

Micro-condition Codes:

a. Word mode:

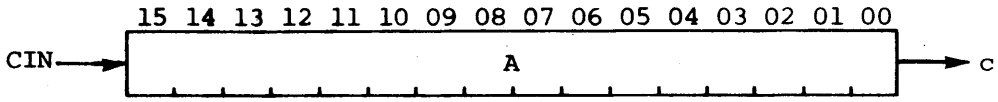
c = (A_{15})
v = $(A_{15})\oplus(A_{14})$
z = 1 if (R)=0;
0 otherwise
n = 1 if (R_{15})=1;
0 otherwise
p = 1 if (R)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

b. Byte mode:

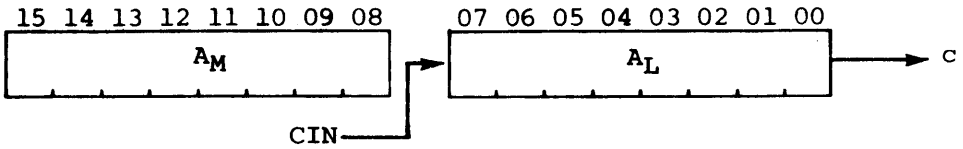
c = (A_{07})
v = $(A_{07})\oplus(A_{06})$
z = 1 if (R_L)=0;
0 otherwise
n = 1 if (R_{07})=1;
0 otherwise
p = 1 if (R_L)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

Logical Open Right Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand is shifted one bit to the right. The value of CIN designated by the MC field is shifted into bit 15. The bit shifted out of bit 00 is the shift carry-out. The result is transferred to the destination.

Byte shifts are the same as word shifts, except that the shift is on the less-significant byte only. The value of CIN is shifted into bit 07. The more significant byte is unmodified. The resulting word is transferred to the destination.

Micro-condition Codes:

a. Word mode:

c = (A_{00})

v = $(A_{15}) \oplus$ CIN

z = 1 if (R)=0;
0 otherwise

n = 1 if (R_{15})=1;
0 otherwise

p = 1 if (R)>0;
0 otherwise

d = 1 if (R_{00})=1;
0 otherwise

b. Byte mode:

c = (A_{00})

v = $(A_{07}) \oplus$ CIN

z = 1 if (R_L)=0;
0 otherwise

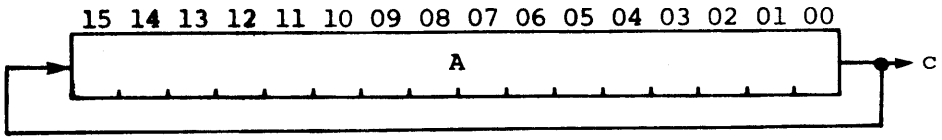
n = 1 if (R_{07})=1;
0 otherwise

p = 1 if (R_L)>0;
0 otherwise

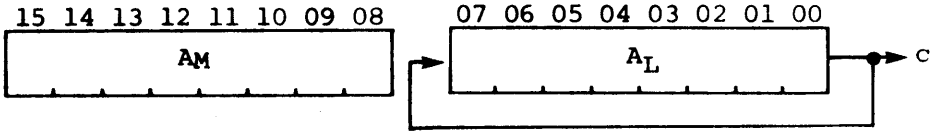
d = 1 if (R_{00})=1;
0 otherwise

Logical Closed Right Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand is shifted right one bit. Bit 00 is the shift carry-out and is also shifted into bit 15. The result is transferred to the destination.

Byte shifts are the same as word shifts, except that the shift is on the less-significant byte only. The carry bit is shifted into bit 07. The more-significant byte is unmodified. The resulting word is transferred to the destination.

Micro-condition Codes:

a. Word mode:

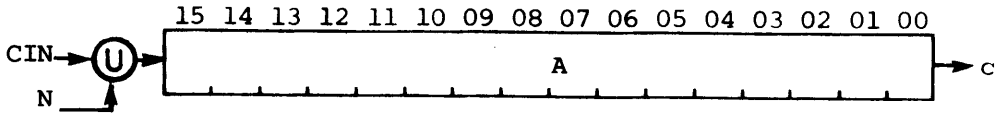
c = (A_{00})
v = $(A_{15})\oplus(A_{00})$
z = 1 if (R)=0;
0 otherwise
n = 1 if (R_{15})=1;
0 otherwise
p = 1 if (R)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

b. Byte mode:

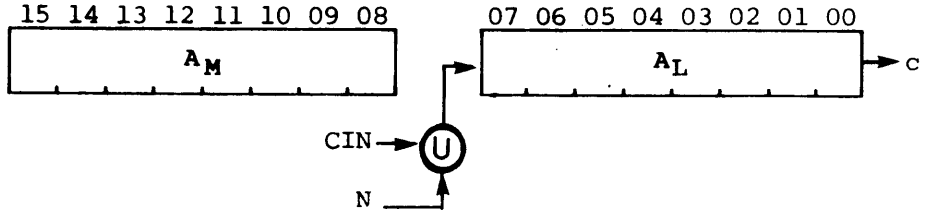
c = (A_{00})
v = $(A_{07})\oplus(A_{00})$
z = 1 if (R_L)=0;
0 otherwise
n = 1 if (R_{07})=1;
0 otherwise
p = 1 if (R_L)>0;
0 otherwise
d = 1 if (R_{00})=1;
0 otherwise

Arithmetic Open Right Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand is shifted right one bit. The state of the negative microstatus bit (N) in MR, ORed with the value of CIN designated by the MC field, is shifted into bit 15. Bit 00 is the shift carry-out. The result is transferred to the destination.

Byte shifts are the same as word shifts, except that the shift is on the less-significant byte only. The value of N ORed with CIN is shifted into bit 07. The more-significant byte is unmodified. The resulting word is transferred to the destination.

Micro-condition Codes:

a. Word mode:

c = (A_{00})

v = $(A_{15}) \oplus [CIN \cup (N)]$

z = 1 if (R)=0;
0 otherwise

n = 1 if (R_{15})=1;
0 otherwise

p = 1 if (R)>0;
0 otherwise

d = 1 if (R_{00})=1;
0 otherwise

b. Byte mode:

c = (A_{00})

v = $(A_{07}) \oplus [CIN \cup (N)]$

z = 1 if (R_L)=0;
0 otherwise

n = 1 if (R_{07})=1;
0 otherwise

p = 1 if (R_L)>0;
0 otherwise

d = 1 if (R_{00})=1;
0 otherwise

Arithmetic Closed Right Shift (Word or Byte).

Description: Same as logical closed right shift.

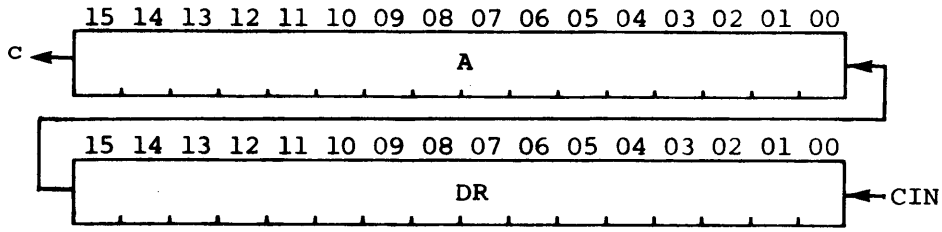
5.3.3.1.2 Double-Precision Shifts. Double-precision shifts involve AU shift elements and DR. When a double-precision shift is specified in the SO field, a DR shift operation specified by the FN field is ignored.

Swap Halves (Word or Byte).

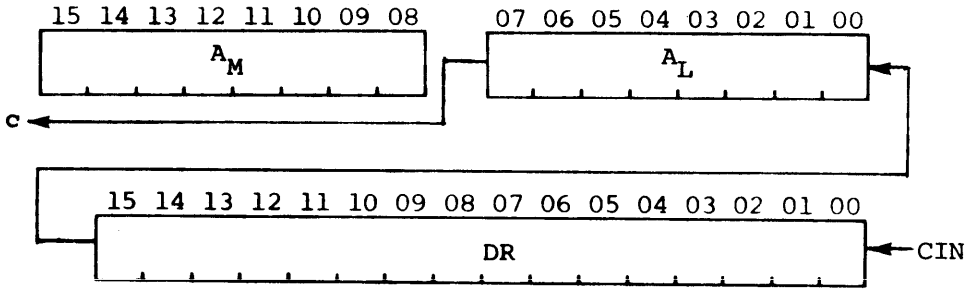
Description: Operation on the A operand and the microcondition codes generated are the same as for single-precision swap. The contents of DR are unmodified.

Logical Open Left Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand and the contents of DR are shifted left one bit. The value of CIN designated by the MC field is shifted into DR bit 00. The DR bit 15 is shifted into the A-operand bit 00. A-operand bit 15 is the shift carry-out. The shifted A-operand result is transferred to the destination. The shifted DR result remains in DR.

Byte shifts are the same as word shifts, except that the A-operand shift is on the less-significant byte only. A-operand bit 07 is the shift carry-out. The more-significant byte is unmodified.

Micro-condition Codes:

a. Word modes:

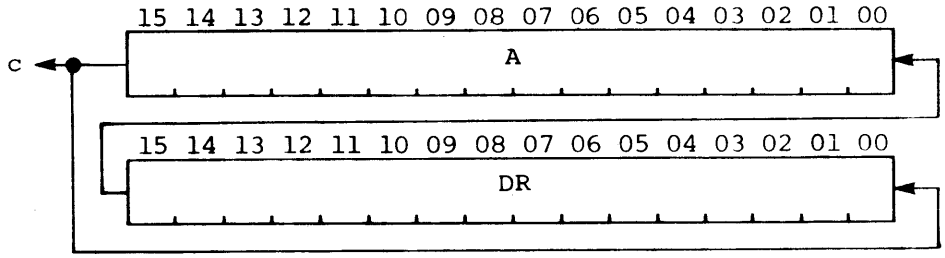
c = (A_{15})
v = $(A_{15})\oplus(A_{14})$
z = 1 if (A)=0;
0 otherwise
n = 1 if (A_{15})=1;
0 otherwise
p = 1 if (A)>0;
0 otherwise
d = 1 if (DR_{15})=1;
0 otherwise

b. Byte modes:

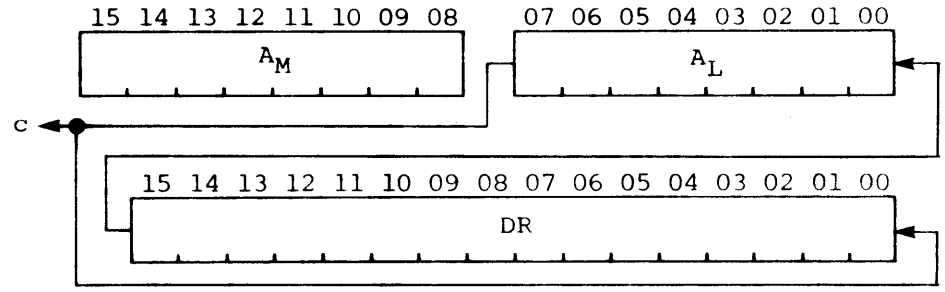
c = (A_{07})
v = $(A_{07})\oplus(A_{06})$
z = 1 if A_L)=0;
0 otherwise
n = 1 if (A_{07})=1;
0 otherwise
p = 1 if (A_L)>0;
0 otherwise
d = 1 if (DR_{15})=1;
0 otherwise

Logical Closed Left Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



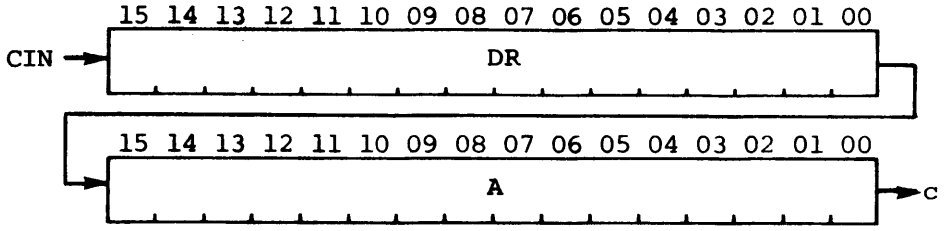
Description: For word shifts, the A operand and the contents of DR are shifted left one bit. DR bit 15 is shifted into A-operand bit 00. A-operand bit 15 is the shift carry-out and is also shifted into DR bit 00. The shifted A-operand result is transferred to the destination. The shifted DR result remains in DR.

Byte shifts are the same as word shifts, except that the A-operand shift is on the less-significant byte only. A-operand bit 07 is the shift carry-out and is also shifted into DR bit 00. The more-significant byte is unmodified.

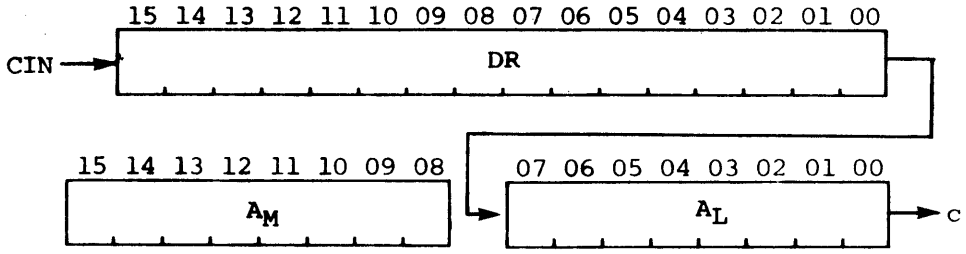
Micro-condition Codes: Same as logical open left shift.

Logical Open Right Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand and the contents of DR are shifted right one bit. The state of CIN designated by the MC field is shifted into DR bit 15. DR bit 00 is shifted into A-operand bit 15. A-operand bit 00 is the shift carry-out. The shifted A-operand result is transferred to the destination. The shifted DR result remains in DR.

Byte shifts are the same as word shifts, except that the A-operand shift is on the less-significant byte only. DR bit 00 is shifted into A-operand bit 07. The more-significant byte is unmodified.

Micro-condition Codes:

a. Word mode:

$c = (A_{00})$

$v = (DR_{00}) \oplus (A_{15})$

z = 1 if (A)=0;
0 otherwise

n = 1 if (DR_{15})=1;
0 otherwise

p = 1 if (A)>0;
0 otherwise

d = 1 if (A_{00})=1;
0 otherwise

b. Byte mode:

$c = (A_{00})$

$v = (DR_{00}) \oplus (A_{07})$

z = 1 if (A_L);
0 otherwise

n = 1 if (DR_{15})=1;
0 otherwise

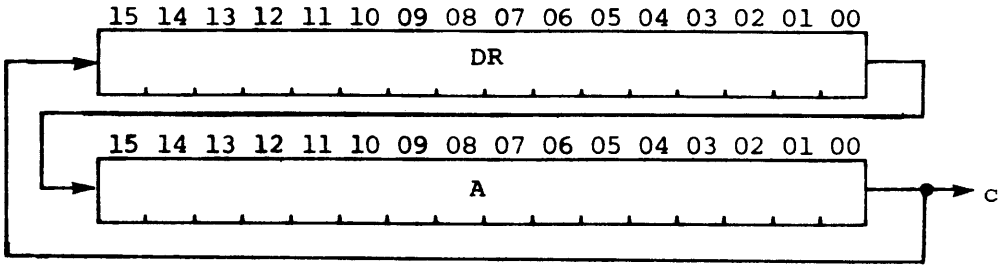
p = 1 if (A_L)>0;
0 otherwise

d = 1 if (A_{00})=1;
0 otherwise

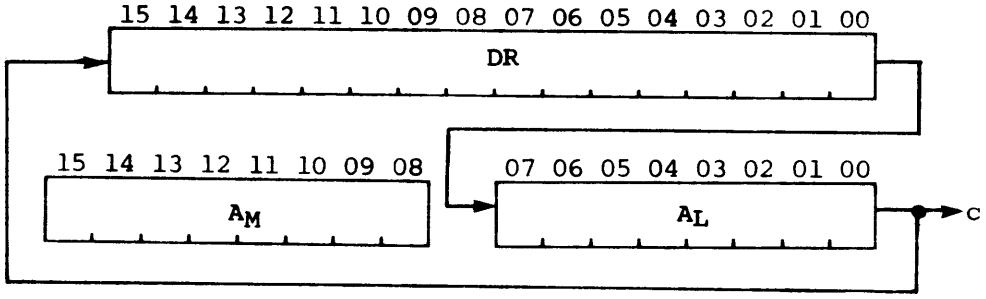
Logical Closed Right Shift (Word or Byte).

Operation:

a. Word mode:



b. Byte mode:



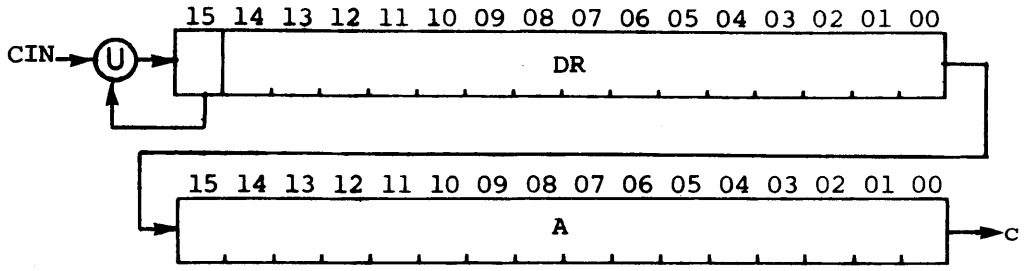
Description: For word shifts, the A operand and the contents of DR are shifted right one bit. DR bit 00 is shifted into A-operand bit 15. A-operand bit 00 is the shift carry-out. The shifted A-operand is transferred to the destination. The shifted DR result remains in DR.

Byte shifts are the same as word shifts, except that the A-operand shift is on the less-significant byte only. DR bit 00 is shifted into A-operand bit 07. The more-significant byte is unmodified.

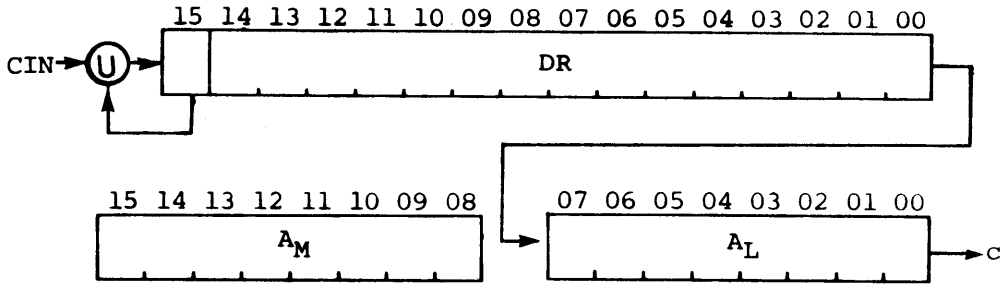
Micro-condition Codes: Same as logical open right shift.

Arithmetic Open Right Shift (Word or Byte).

Operation: a. Word mode:



b. Byte mode:



Description: For word shifts, the A operand and the contents of DR are shifted right one bit. DR bit 15, ORed with the state of CIN designated by the MC field, is shifted into DR bit 15. DR bit 00 is shifted into A-operand bit 15. A-operand bit 00 is the shift carry-out. The shifted A-operand result is transferred to the destination. The shifted DR result remains in DR.

Byte shifts are the same as word shifts, except that the A-operand shift is on the less-significant byte only. DR bit 00 is shifted into A-operand bit 07. The more-significant byte is unmodified.

Micro-condition Codes: Same as logical open right shift.

Arithmetic Closed Right Shift (Word or Byte).

Description: Same as logical closed right shift.

5.3.3.2 Multiply Step

The MUS microcommand is a specialized version of the Shift microcommand with an automatic iterative repeat that permits high-speed implementation of a Multiply instruction. The average execution time is 300 nanoseconds per bit plus the additional time required to preformat the multiplier and multiplicand, determine the sign of the product and format the final result. No additional hardware is required for the high-speed multiply function, since all operations are implemented in control memory.

Mnemonic: MUS $19

Microcommand Type: Special branch.

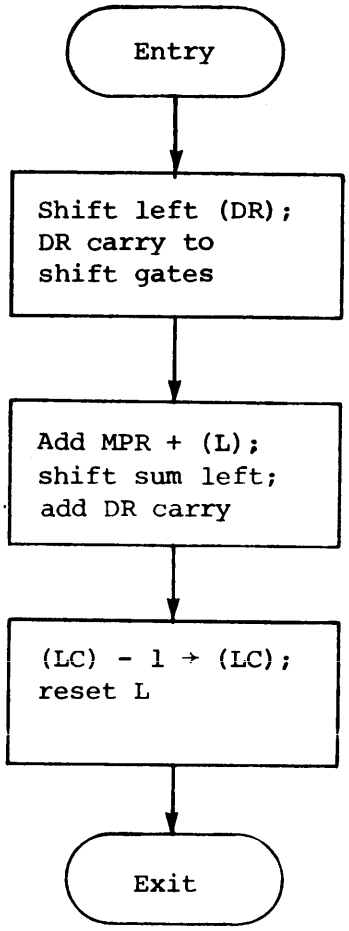
Description: The MUS microcommand provides a set of simultaneous add, shift and test operations involving a file register containing the multiplier (MPR) plus DR, LC and the state of the next multiplier digit. The microcommand is automatically repeated until (LC)=0. For each ONE in the multiplier, a branch is made to a microcommand that adds the multiplicand (MPD) to DR. This permits complete execution of multiply steps in one clock cycle for a ZERO multiplier digit and three clock cycles for a ONE multiplier digit. The MUS microcommand is used for multiplication of two 16-bit operands with a resulting 32-bit product.

Registers Used:

a. MPR in a file register location designated by the AO field of the MUS microcommand. This file register location contains the more-significant half of the product at the end of the complete multiplication.
b. MCD in a file register location designated by a separate microcommand that adds MCD to the partial product.
c. DR, which accumulates MCD additions to the partial product and contains the less-significant half of the product at the end of the complete multiplication.
d. LC, which counts the number of MUS iterations performed.
e. The link microstatus bit used to propagate carries from the less-significant half to the more-significant half of the partial product.

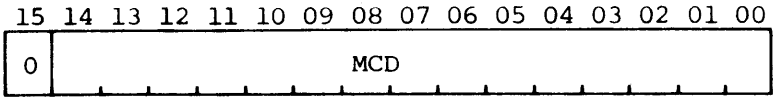
SO Field: The SO field must be programmed for a double-precision, logical open left shift (bits 15 to 12 = 0110), as specified in the Shift microcommand description (Table 5-4).

Operation: The following operations are executed simultaneously by the MUS microcommand.

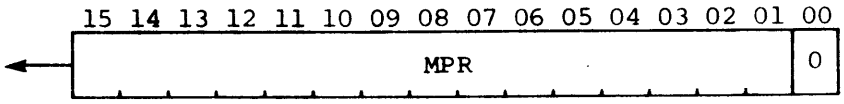


Procedure:

1. Convert MCD to a positive number:



2. Convert MPR to a positive number, shift left and test MPR relative to zero:

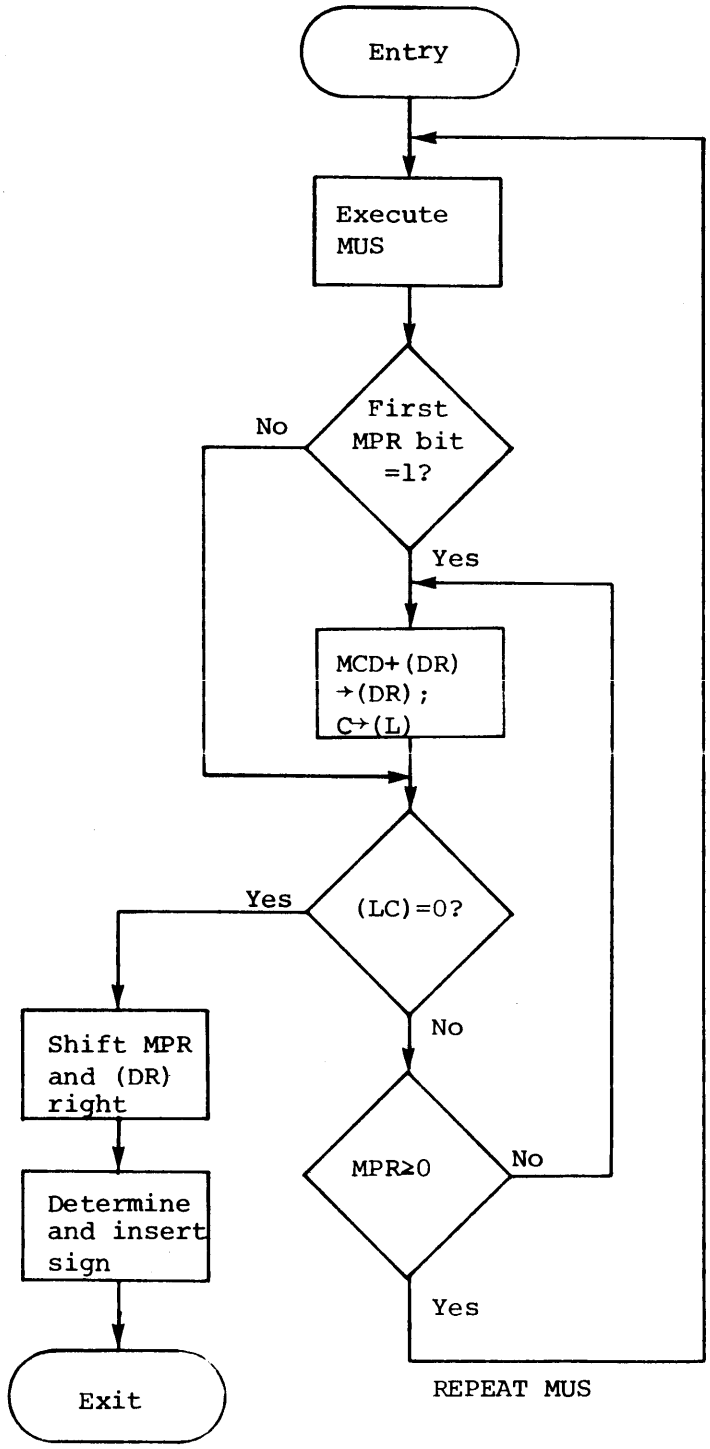


3. Initial conditions:
   (DR)=0.
   (L) =0.
   (LC)=15 (for a 16-bit multiplication).
4. Program MUS control fields as follows:
   SB = $10 (dynamic branch < 0).
   OP = $19 (MUS).
   DN = MPR address.
   NX = $3 (inhibit next microcommand, if branch).
   AO = MPR address.
   MC = $6 (add and update L).
   MX = $0 (no operation).
   SO = $6 (double-precision logical open left shift).
   BF = location of MUS-1.

   The symoblic microassembler automatically sets up all fields except DN and AO.

5. The final double-length result the MPR file register and DR must be shifted right one bit after the last iteration. This can be performed using the standard SHF microcommand programmed for a double-precision logical open right shift. The sign of the product must also be determined and inserted.

6. The basic microcommand sequence is illustrated below:



Micro-condition Codes:

c = AU shift carry
v = shift overflow
z = 1 if AU shift result = 0; 0 otherwise
n = 1 if AU shift result most-significant bit = 0; 0 otherwise.
p = 1 if AU shift result > 0; 0 otherwise
d = 1 if AU shift result least significant bit = 0; 0 otherwise.

Example:

Multiply the following four-bit numbers (all registers assumed to be four bits):

MPR = 0101
MCD - 0111

MPR	L	DR	LC	Explanation	
0101	0	0000	3	Initial condition.	
1010		____		Shift MPR left; test MPR<0.	
	0	0111		Add MCD to (DR).	
0	←			Add (L) to MPR.	
1010			2		MUS
0100	0	1110		Shift MPR and (DL) left	
0	←				
0100			1		MUS
1001	0	1100			
	1	0011		Add MCD to (DR).	
1	←				
1010			0	Exit, (LC)=0.	MUS
0100		0110			
0010		0011		Shift MPR and (DR) right for final product.	

product

5.3.3.3 Divide Step

The DVS microcommand is a specialized version of the subtract operation (conditional) with an automatic iterative repeat that permits high-speed implementation of a Divide instruction. The fixed execution time is two clock cycles per bit plus the time required to preformat the divisor and dividend, check for overflow, determine the sign of the quotient and format the final result. No additional hardware is required for the high-speed division, since all operations are implemented in control memory.

Mnemonic: DVS $1A

Microcommand Type: Special branch.

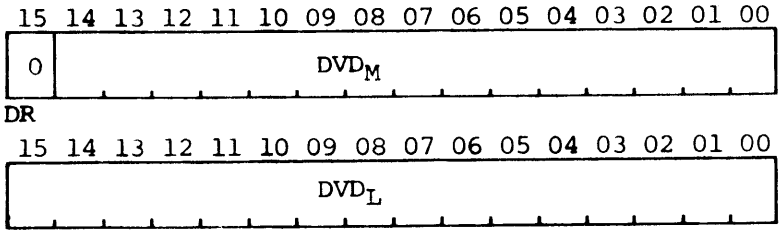
Operation: (R)=(A)+(B)+1
If c=1, (R)→(A)

Description: The DVS microcommand executes a two's complement addition of the divisor (DVR) to the more-significant word of a double-precision dividend (DVD). If a carry-out is generated by the addition, the result replaces the more-significant word of DVD; otherwise, DVD is unchanged.

The carry-out must be saved in the L microstatus bit. The microcommand is used in conjunction with a double-length left shift of the A operand and DR on each iteration, with the carry-out saved in L shifted into DR. DVS can be automatically repeated using LC. The result is a single-length quotient with a single-length remainder.

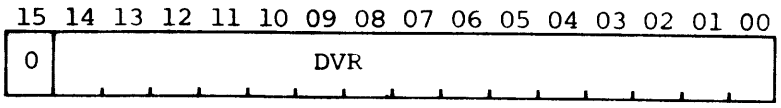
Registers Used:
a. Dividend more-significant word (DVD_M) in a file register designated by the AO field of the DVS microcommand. This register contains the remainder at the end of the complete division operation.
b. Dividend less-significant word (DVD_L) in DR.
c. DVR in a file register designated by the BO field of the DVS microcommand. This register contains the quotient at the end of the complete division operation.
d. LC, which counts the number of iterations performed.
e. The link microstatus bit used to propagate quotient bits into DR.

Procedure: 1. Convert DVD to a 31-bit positive number:



A Operand

2. Convert DVR to a positive number:

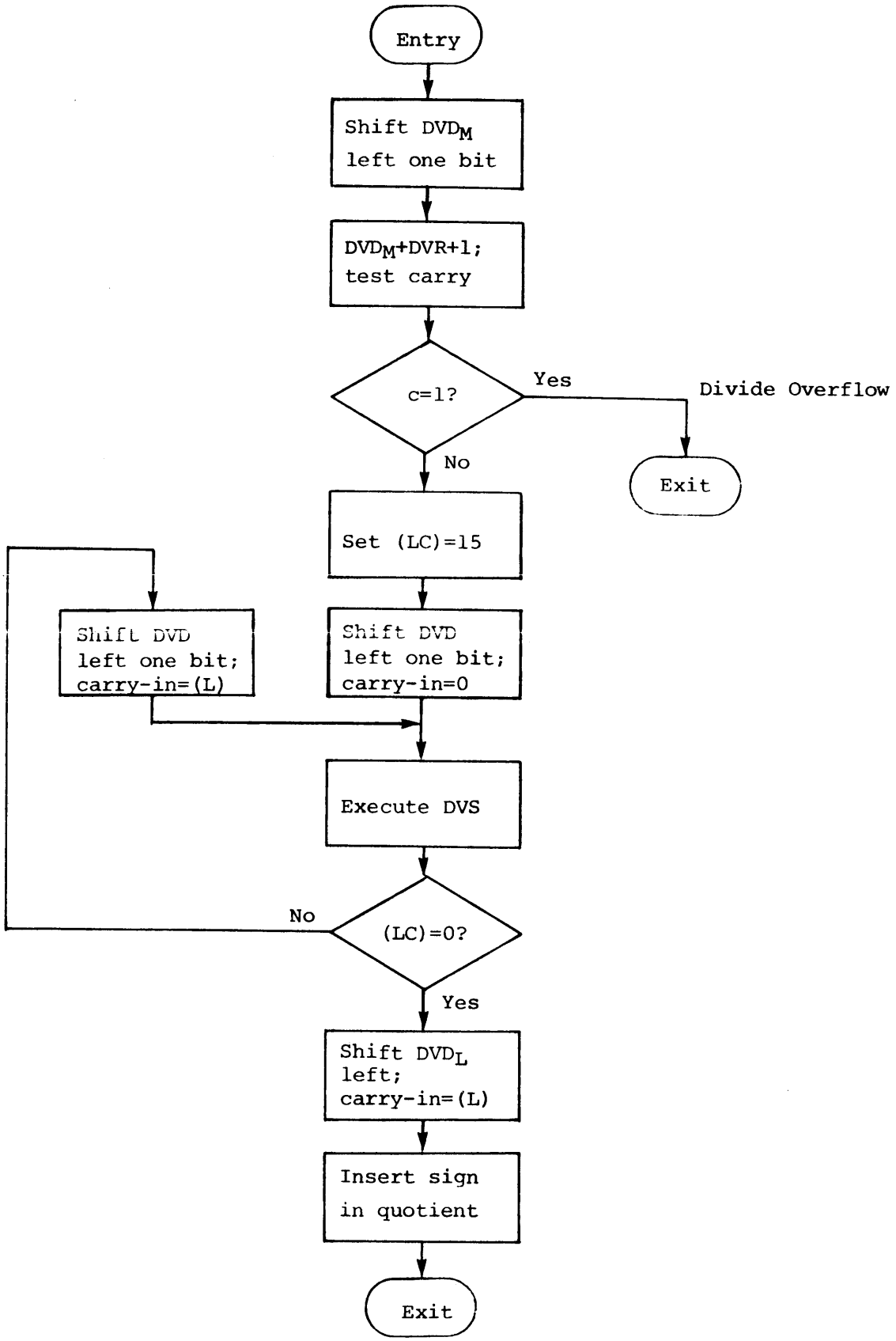


B Operand

3. Test for DVR ≤ 2 x DVD. If true, set overflow and exit.
4. If no overflow, shift DVD left and set initial conditions:
   (1)=0.
   (LC)=15.
5. Program DVS control fields as follows:
   SB = $03 (branch if (LC) = 0 before decrementing).
   OP = $1A (DVS).
   DN = DVD_L.
   NX = $1 (execute next microcommand, if branch).
   AO = DVD_M).
   MC = $5 (modify link status; carry input = 1).
   MX = $0 (no operation).
   BO = DVR.
   BF = location of DVS.

   The symoblic microassembler automatically sets up all fields except DN, AO and BO.
6. DVS is used in conjunction with a double-length left shift on the A operand and DR, with L added to DR.
7. The final double-length result must be left shifted one bit after the final iteration. The quotient is in DR and the remainder in the A-operand source. The sign of the quotient is determined and set separately.

8. The basic microcommand sequence is illustrated below:



Example: Divide the following numbers (all registers are assumed to be four bits):

DVD = 0011 0110
DVR = 0111

L	DVD	DR(DVD_L)	LC	Explanation	
0	0011	0110	3	Initial condition.	
0	0110	1100	3	Shift DVD left.	
	1001			Add DVR two's complement (no DVD modify).	
	1111			No overflow, continue.	
	1101	1000	3	Shift DVD left.	DVS
	1001			Add DVR two's complement.	
1	0110	1000	2	Modify DVD_M. Test LC.	
	1101	0001	2	Shift DVD left. Add L.	
	1001				DVS
1	0110	0001	1		
	1100	0011	1	Shift (DVD) left. Add L.	
	1001				DVS
1	0101	0011	0	Exit, (LC)=0.	
	0101	0111		Shift DVD_L left. Add L.	

Remainder Quotient

5.3.3.4 Test Bit

The TSB microcommand provides the ability to test and conditionally branch on the state of a specified bit in the A operand. The microcommand cannot be a skip type (the K bit, 47, is ignored) and the BF field is always treated as a branch address. The SB field can also specify a separate branch condition. If either the bit test or the SB-field condition is met, the branch occurs. This provides considerable flexibility in performing multiple test operations at high speed.

Mnemonic: TSB $1B

Microcommand Type: Special branch.

Description: The A operand is transferred, unmodified, to the destination. The state of the A-operand bit specified by the SO field is tested. If the bit test condition is met (as specified by the T bit, 43) a branch is made to the location given in the BF field. The SB field can specify an additional branch condition. If either the bit test or the SB condition is met, the branch occurs.

SB Field:

a. The K bit (47) is ignored.
b. The T bit (43) specifies the ONE or ZERO state of both the bit test and the SB-field test (i.e., both must test the same state).
c. The SB-field test conditions are given below. The normal unconditional branch condition is treated as a no-branch. This no-branch must be programmed if only the bit test condition is to be tested.

Micro-condition Codes: Same as arithmetic addition (Table 5-3).

SB	Test Condition
$0	loop count equals zero
$1	carry
$2	overflow
$3	zero
$4	negative
$5	positive
$6	odd
$7	unconditional branch (treated as no-branch in TSB)

5.3.3.5 Modify Macrostatus (Optional)

The CDP-XI/00 contains, in addition to the microlevel status in MS, a macrostatus register (PS) that stores processor macrolevel conditions, including link, overflow, negative and zero as well as other information, on the states of the emulated computer. These processor status conditions are generated at intermediate times by the emulation microroutines that must be transferred to PS by microcommand. Since PS update can differ for many types of instructions being emulated, the update operation can add an excessive number of microcommands to the emualation microroutines.

To provide fast processor status update, the emulate table can be programmed to generate a set of PS-update control bits that are specific to the instruction contained in IR. When the MMS microcommand is executed, these control bits steer the contents of MB directly to the proper PS location. In this way, the microprogram generates proper PS values for each emulated instruction using only one clock step.

Since each emulated computer requires a different treatment of PS values, the emulate table associated with MMS is unique. In some cases, MMS is not needed at all to meet overall emulation speed objectives. For this reason, the MMS is considered an optional microcommand that can be omitted, or tailored for a specific emulation task.

Mnemonic: MMS $1C

Microcommand Type: Special branch or special skip.

Description: The A operand is transferred to the destination. Any or all of the least-significant four bits of the A operand can also be transferred to the corresponding least-significant four bits of PS, if specified by the microcommand. The contents of IR are translated into a set of update functions that specify a modification of the least-significant four bits of PS.

The update functions are contained in the emulate table. The update functions that can be specified individually for PS bits 03 to 00 are:

PS Update Function	PS03	PS02	PS01	PS00
No change	$(PS_{03})\rightarrow(PS_{03})$	$(PS_{02})\rightarrow(PS_{02})$	$(PS_{01})\rightarrow(PS_{01})$	$(PS_{00})\rightarrow(PS_{00})$
Reset	$0\rightarrow(PS_{03})$	$0\rightarrow(PS_{02})$	$0\rightarrow(PS_{01})$	$0\rightarrow(PS_{00})$
Set	$1\rightarrow(PS_{03})$	$1\rightarrow(PS_{02})$	$1\rightarrow(PS_{01})$	$1\rightarrow(PS_{00})$
Conditional	$(A_{03})\rightarrow(PS_{03})$	$(A_{02})\rightarrow(PS_{02})$	$(A_{01})\rightarrow(PS_{01})$	$(A_{00})\rightarrow(PS_{00})$

The update functions permit each PS bit to be left unaltered, unconditionally reset, unconditionally set or modified by the contents of the corresponding four bits of the A operand, which is routed via MB. If MS is selected as the A-operand source, the MMS microcommand can transfer the L, V, N and Z microstatus bits directly to PS.

Micro-condition Codes:

Same as arithmetic addition (Table 5-3).

5.3.3.6 Conditional Memory Access (Optional)

For emulation of a set of instructions involving one or more operands, it is usually desirable to read some operands from memory in a read/restore mode and others in a read/modify/write mode. The read/restore mode is associated with operands that are not modified by the instruction. Examples are:

a. Load (memory to hardware register).
b. Add (memory to hardware register).
c. Compare (memory with hardware register).

For high emulation speed, the address mode and operand fetch operations are generally executed before the specific operation is determined, so the memory access mode is not known at the time the operand fetch cycle is initiated. If a read/write mode is used in all cases, an extra memory cycle is required to restore the modified operand. Use of a read/modify/write operation saves both memory and processor time.

The CMA microcommand uses the emulate table to generate a control signal that specifies whether the memory access is to be read/restore or read/modify/write. This is determined by the contents of IR, which holds the current instruction being emulated. Since the table is unique for each emulation, and in some cases may not be required, the CMA microcommand is considered optional.

Mnemonic: CMA $1D

Microcommand Type: Special skip or special branch.

Description: The A operand is transferred to the destination. The CMA microcommand automatically generates either a memory read/restore or a memory read/modify/write operation on the MACROBUS. The type of operation is determined by the state of the conditional memory access control bit from the emulate table. The location of the memory word is specified by the contents of the A-operand source. The operation is performed in the word or byte mode, depending on the state of the I/O byte control bit from the emulate table. The word (or byte) read from memory is stored in RR when received.

Micro-condition Codes: Same as arithmetic addition (Table 5-3).

Programming: In a special skip-type microcommand, the FN field can generally designate a memory access operation; however, any FN field memory access operation is overridden by the conditional memory access operation specified by the OP field.

5.3.3.7 Decode

In emulation microroutines, it is desirable to have a means to modify specific bit fields in a given microcommand, based on the particular instruction being emulated. For example, an add and a subtract microroutine may differ only in that the operands are added or subtracted. By modifying the OP field of the arithmetic microcommand, a common routine can be used. Another example is entering a particular file register address based on a field in the instruction. The DCD microcommand permits this type of operation to be accomplished directly through use of a decode table that modifies specified bits in the microcommand following the DCD microcommand. The table is set up for the specific emulation and, in some cases, may not be needed. For this reason, the DCD is considered an optional microcommand.

Mnemonic: DCD $1F

Microcommand Type: Special skip or special branch.

Description: A zero word is placed on MB. The DCD microcommand selects a 16-bit modifier from the decode table. This word modifies a specified set of bits in the least-significant 16 bits of the next microcommand read from control memory (prior to execution). The modifier and bit fields to be modified are selected using the AO and BO fields of DCD and the contents of ER.

The AO and BO fields of DCD are used as follows:

a. Bits 28 and 27 of the AO field select one of four groups of four bits each in ER. The ER bit group selected is taken as the least-significant four bits of an eight-bit address to the decode table.
b. Bits 26 to 24 of the AO field select one of eight possible field modification patterns for the next microcommand read from CM.
c. The BO field is taken as the most-significant four bits of the eight-bit address to the decode table. The 16-bit modifier word fetched from the decode table is ANDed with the least-significant 16 bits in the next microcommand read from CM before it is transferred to CR for execution.

Microcondition Codes: Save arithmetic addition. Since the MB result is always zero, the microcondition codes are:
c = 0, v = 0, z = 1, n = 0, p = 0, d = 0.

SECTION 6
INSTALLATION

6.1 GENERAL

When used as part of a system, the CDP-XI/00 processor boards will have been installed in a CDP-XI processor chassis and tested by Cal Data prior to shipment.

6.2 UNPACKING AND INSPECTION

The processor chassis (with boards installed) is shipped in a padded shipping container for protection during transportation. This container can be saved for future use if the unit is returned for repair or reshipped separately from the associated system.

The following steps are recommended for unpacking and inspecting the processor:

1. Prior to opening, inspect the box for obvious damage.
2. Cut the packing tape, open the box and remove the chassis. Next, remove the plastic wrapper and inspect the chassis and boards for physical damage.

It is important to note immediately any physical damage that might have resulted from shipment. The carrier should be notified of such damage and given the opportunity to inspect the unit and container. This helps establish the validity of any claims for shipping insurance.

6.3 INSTALLATION

Prior to operation of the processor, the chassis must be properly installed in a suitable rack or cabinet, all required boards and the power supply must be installed in the chassis and all necessary cables must be routed and connected.

The following paragraphs provide step-by-step directions for all phases of installation of the CDP-XI chassis, processor and macropanel boards. Power supply, memory and peripheral device installation and cabling is described in the separate manuals for those units.

6.3.1 Handling

The chassis and boards are designed to withstand all normal shock and vibration encountered in shipping and when installed in a computer system. While not fragile devices, they should be handled with reasonable care to avoid damage that might result in operational failure.

Avoid bending components when handling any assembly. To prevent oxides from forming on gold plating, do not touch connector pins.

6.3.2 Chassis Installation

The processor chassis is delivered with the top and bottom covers, backplane and fans in place. It slides into a standard 10-1/2 inch (26.7 cm) position of a 19-inch (48 cm) wide RETMA rack.

After placing the chassis in the rack, slide it out to the final front stop, which will extend the entire chassis beyond the front of the rack. Remove the top cover (six screws). Then remove the bottom cover (six screws) in order to install the power supply in the chassis. Directions for installing the power supply are given in the power supply manual.

After the power supply has been installed, replace the bottom cover and install the processor and any other required boards according to the directions given in paragraphs 6.3.3 and 6.3.4. Install and route any needed peripheral cables. Replace the chassis top cover, slide the chassis back into the cabinet, install the two holding screws, and snap on the front panel cover to complete the installation.

6.3.3 Board Installation

Always ensure that system power is OFF when installing or removing boards. Insert and remove each board slowly and carefully so that it does not make contact with adjacent boards. Never use components as finger grips; use the grip areas at the corners of the boards.

Chassis slots 1 to 5 are closely spaced for the installation of processor boards (Figure 2-8). The macropanel board goes in slot 1, engine boards 1 and 2 in slots 2 and 3, respectively, I/O control board in slot 4, and the emulate board in slot 5.

Chassis slots 6 to 12 are on wider centers for option, memory or I/O controller boards. If the optional memory management unit is present, it must be installed in slot 6. Any other processor option can be installed in either slot 6 or 7. Memory modules are installed in a contiguous group of slots, beginning with the first available slot (6, 7 or 8). I/O controllers are placed behind these. Details on the installation of options, memory and I/O controllers are given in the applicable individual documentation.

A terminator or MACROBUS cable is installed in the A and B connectors of the first vacant slot, which can be slot 13 of the standard 12-slot backplane.

Installation of the macropanel board is described in paragraph 6.3.4. Install the processor boards with components toward the front of the chassis. Check each board for proper orientation before attempting to install it. Because the connectors are keyed, excessive force applied to a reversed board can result in connector damage. Make sure that the board is completely and evenly seated.

When the processor boards are installed in the chassis, connect the two small processor interconnection boards by pressing them firmly but carefully over connectors J1 and J2 on the tops of the processor boards in slots 2 to 5. There is no connection to the macropanel in slot 1 from these boards.

6.3.4 Macropanel Installation

To install the macropanel board, grasp the lower edge of the panel front cover (and bezel), snap it forward away from the chassis, and lift it up off the upper front rail of the chassis.

If slot 2 is not yet occupied, the macropanel board can be inserted through the top of the chassis into the slot 1 connectors, with switches and indicators facing the front of the chassis. If slot 2 is occupied, remove the two small processor interconnection boards from the tops of the boards in slots 2 to 5, insert the top of the macropanel board through the front of the chassis, carefully raising it behind and above the upper front rail far enough so that the lower edge of the board can be seated in the connectors. Attach the board to the macropanel mounting tabs (one on each side) with the nylon screws provided. If applicable, replace the two small interconnection boards.

After the chassis has been pushed back into the rack and the two holding screws installed, align the front panel cover on the upper front rail so that it swings down over the macropanel switches and indicators properly. Snap the cover into place by pressing on the lower edge.

Because the macropanel is treated as an I/O device by the system, it can be installed in any chassis slot with MACROBUS connectors. An extender board is available to allow the macropanel to sit above the top of the standard size circuit boards. This feature is particularly useful during maintenance operations, when the first slot of the chassis is used to hold a board under test (for easy signal tracing). At the same time the macropanel can be mounted within easy reach in any vacant slot.

6.4 CABLING

The processor itself has no associated cabling. All electrical connections between the processor and other system components are made via the backplane. Connections between and among processor boards are via the two small processor interconnection boards.

SECTION 7
MAINTENANCE

(In work)

APPENDIX A
MICROPROCESSOR ARITHMETIC

A.1 NUMBER REPRESENTATION

In the CDP-XI/00, the AU is implemented to perform both addition and subtraction internally (as opposed to complement addition for the subtraction function). Hence, the dynamic arithmetic condition codes generated (carry-out and overflow) and the function of the carry-in (CIN) to the AU depend on whether addition or subtraction is performed. Arithmetic operations assume the use of the two's complement representation for negative numbers in the processor, with the state of the most-significant bit representing the sign of the number. The 16-bit single-precision number range of the CDP-XI/00 is therefore:

Binary	Hex	Decimal
0111111111111111	$7FFF	$2^{15} - 1 = 32,767$
.	.	.
.	.	.
.	.	.
0000000000000001	$0001	1
0000000000000000	$0000	0
1111111111111111	$FFFF	-1
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
1000000000000000	$8000	$2^{-15} = -32,768$

To form the two's complement of a binary number, perform:

$$-|X| = \overline{X} + 1$$

where X is the logical (or one's) complement of the binary number. For example:

```
X = 5 = 0101
X̄     = 1010 (one's complement)
+1    =+0001
-X    = 1011 (two's complement)
```

A.2 ADDITION

If all negative numbers are represented in two's complement form, then the result of any addition generates the proper result, regardless of the sign of the two operands.

Examples:

```
  (+4) =  0100    (-4) =  1100
+(+2) = +0010   +(-2) = +1110
=(+6) =  0110   =(-6) = (a) 1010

  (+4) =  0100    (-4) =  1100
+(-2) = +1110   +(+2) = +0010
=(+2) = (a)0010 =(-2) =  1110
```

The notation (a) indicates that a carry output is generated by AU. This carry-out is generally of no significance in addition unless the two operands represent something other than the most-significant bits of a multiple-precision set of numbers. In such case, the carry-out bit can be saved as the link (L) bit and added to the next most-significant set of bits when the next step of the multiple-precision addition is performed. For example, suppose that the following two eight-bit numbers are added using a four-bit adder:

```
                 Step 2     Step 1
    (+44) =      0100       1100
  +(-23) =     +1110       1001
               (a)0000   (a) 0101
Add link =     0001 ←
  =(-21)       0001       0101
```

In the previous example, none of the additions resulted in an arithmetic overflow (i.e., all results are within the maximum number range possible, which for the four-bit numbers is $2^7-1 < \text{range} < 2^{-7}$). An overflow occurs if two positive numbers are added with a sum greater than seven:

```
  (+5) = 0101
+(+4) = 0100
  (-7) = 1001   (overflow)
```

The negative seven is an incorrect result, and the overflow is determined by a change of sign to negative when the two positive operands are added. A carry-out is not generated.

The carry and overflow condition order for addition are determined in the CDP-XI/00 by:

$$c = \left[(A_{15}) \cap (\overline{R_{15}})\right] \cup \left[(B_{15}) \cap (\overline{R_{15}})\right] \cup \left[(A_{15}) \cap (B_{15})\right]$$

$$v = \left[(A_{15}) \cap (B_{15}) \cap (\overline{R_{15}})\right] \cup \left[(\overline{A_{15}}) \cap (\overline{B_{15}}) \cap (R_{15})\right]$$

where A_{15}, B_{15} and R_{15} are the most-significant bits of the A operand, B operand and result, respectively. The c and v microcondition codes can be stored in the microstatus register (MS) L and V bits, respectively, using the MC field of the microcommand.

In the CDP-XI/00, the carry input (CIN) to AU can also be specified by the MC field of the microcommand. The states can be programmed as:
CIN = 1, CIN = 0 or CIN = (L).

The ADD microcommand is (A)+(B)+CIN, where CIN is the carry-in under MC field control. Thus, it is possible to add the fixed constants ONE or ZERO to the result, or to add the state of the L bit (which can contain the carry propagation for multiple-precision addition, for example).

A.3 SUBTRACTION

The CDP-XI/00 performs true binary subtraction as well as addition. This provides considerably greater flexibility in implementing the arithmetic microcommands than would the usual use of complement addition.

Examples:

```
  (+4) =   0100        (-4) =   1100
-(+2) =  -0010      -(-2) =  -1110
=(+2) =   0010      =(-2) =(a)1110

  (+4) =   0100        (-4) =   1100
-(-2) =  -1110      -(+2) =  -0010
=(+6) =(a)0110      =(-6) =   1010
```

The notation (a) in this case indicates that a borrow output is generated by the subtractor. The borrow-out is of significance only if the two operands represent something other than the most-significant bits of a multiple-precision set of numbers. In such a case, the borrow-out bit can be saved as the link (L) bit and then subtracted from the result of subtracting the next-most-significant bits. For example, suppose that the following two eight-bit numbers are subtracted using a seven-bit subtractor:

```
                (+60  = 0011      1100
              -(+30) =-0001  (a)  1110
Subtraction link =-0001<--------  1110
              =(+30)    0001      1110
```

Overflow results from subtraction, as from addition, when the result is outside the range of the number system ($2^7-1 < \text{result} < 2^{-7}$ for a four-bit range). The borrow and overflow condition codes for subtraction are determined in the CDP-XI/00 by:

$$c = [(\overline{A_{15}}) \cap (R_{15})] \cup [(\overline{A_{15}}) \cap (B_{15})] \cup [(B_{15}) \cap (R_{15})]$$

$$v = [(\overline{A_{15}}) \cap (B_{15}) \cap (R_{15})] \cup [(A_{15}) \cap (\overline{B_{15}}) \cap (\overline{R_{15}})]$$

The borrow is designated as c in the processor. The condition codes can be stored in the microstatus register L and V bits, respectively, using the MC field of the microcommand.

In the CDP-XI/00, the borrow input (also CIN) to AU can also be specified by the MC field of the microcommand. The states can be programmed as:

CIN = 1, CIN = 0 or CIN = (L).

The SUB microcommand is (A)-(B)-CIN, where CIN is the borrow-in under MC field control.

APPENDIX B
FIXED MEMORY ASSIGNMENTS

System interrupt vectors (two words per vector) are given in Table B-1. Only those vectors used by the CDP-XI/00 processor and standard Cal Data options are given. Other vector locations are reserved. Users should observe these assignments if full software compatibility is to be retained.

Standard I/O device MACROBUS addresses are given in Table B-2. Assignments for the CDP-XI/00 processor and standard Cal Data options are given.

Table B-1. Interrupt Vectors

Octal Address	Use
000	Reserved
004	MACROBUS timeout error
010	Reserved instruction vector
014	Debug trap vector
024	Power failure trap vector
034	"Trap" trap vector
060	Serial channel in (BR4)
064	Serial channel out (BR4)
070	High-speed reader (BR4)
074	High-speed punch (BR4)
100	Line-frequency clock (BR6)
200	Line printer (BR4)
244	Floating-point error Memory-management abort
254	Macropanel interrupt
300	Start of floating vectors

Table B-2. Standard MACROBUS Device Addresses

Octal Address	Use	Octal Address	Use
777776	Processor status, PS	777560	Teleprinter IN CSR
777774	Stack-limit register	777556	Punch DBR
777772	Other processor registers	777554	Punch CSR
777717		777552	Reader DBR
777716		777550	Reader CSR
777715		777546	Line-frequency clock
777714		777540	Macropanel control switches
777713		777516	Line printer DBR
777712		777514	Line printer CSR
777711		777512	Line printer XX
777710		777510	Line printer XX
777707	FR7 register	777166	Card reader
777706	FR6 register	777164	
777705	FR5 register	777162	
777704	FR4 register	777160	
777703	FR3 register	776676 to 776500	Multiple teleprinter (First unit is at 0776500)
777702	FR2 register		
777701	FR1 register		
777700	FR0 register		
777656	UPAR #7	772356	EPAR #7
777654	UPAR #6	772354	EPAR #6
777652	UPAR #5	772352	EPAR #5
777650	UPAR #4	772350	EPAR #4
777646	UPAR #3	772346	EPAR #3
777644	UPAR #2	772344	EPAR #2
777642	UPAR #1	772342	EPAR #1
777640	UPAR #0	772340	EPAR #0
777616	UPDR #7	772316	EPDR #7
777614	UPDR #6	772314	EPDR #6
777612	UPDR #5	772312	EPDR #5
777610	UPDR #4	772310	EPDR #4
777606	UPDR #3	772306	EPDR #3
777604	UPDR #2	772304	EPDR #2
777602	UPDR #1	772302	EPDR #1
777600	UPDR #0	772300	EPDR #0
777576	MMU status register #2	760000	0760000 reserved for diagnostics
777572	MMU status register #0	764000	User addresses start here and upward to 0767750
777570	Macropanel switch register		
777566	Teleprinter OUT DBR		
777564	Teleprinter OUT CSR		
777562	Teleprinter IN DBR		

MMU* (User)

MMU* (EXECUTIVE)

*Cannot be accessed by a MACROBUS device

APPENDIX C
MACROBUS PIN ASSIGNMENTS

Signal	Connector	Connector	Signal
INIT L	AA1	AA2	+5 V
INTR L	AB1	AB2	GND
D00 L	AC1	AC2	GND
D02 L	AD1	AD2	D01 L
D04 L	AE1	AE2	D03 L
D06 L	AF1	AF2	D05 L
D08 L	AH1	AH2	D07 L
D10 L	AJ1	AJ2	D09 L
D12 L	AK1	AK2	D11 L
D14 L	AL1	AL2	D13 L
PA L	AM1	AM2	D15 L
GND	AN1	AN2	PB L
GND	AP1	AP2	BBSY L
GND	AR1	AR2	SACK L
GND	AS1	AS2	NPR L
GND	AT1	AT2	BR7 L
NPG H	AU1	AU2	BR6 L
BG7 H	AV1	AV2	GND
BG6 H	BA1	BA2	+5 V
BG5 H	BB1	BB2	GND
BR5 L	BC1	BC2	GND
GND	BD1	BD2	BR4 L
GND	BE1	BE2	BG4 L
ACLO L	BF1	BF2	DCLO L
A01 L	BH1	BH2	A00 L
A03 L	BJ1	BJ2	A02 L
A05 L	BK1	BK2	A04 L
A07 L	BL1	BL2	A06 L
A09 L	BM1	BM2	A08 L
A11 L	BN1	BN2	A10 L
A13 L	BP1	BP2	A12 L
A15 L	BR1	BR2	A14 L
A17 L	BS1	BS2	A16 L
GND	BT1	BT2	C1 L
SSYN L	BU1	BU2	C0 L
MSYN L	BV1	BV2	GND

Power	
Signal	Connector
+5 V	DA2
+5 V	FA2
-15 V	DB2
-15 V	FB2
GND	DC2
GND	FC2
GND	DT1
GND	FT1

Bus Grant	
Signal	Connector
BG7 IN	DK2
BG7 OUT	DL2
BG6 IN	DM2
BG6 OUT	DN2
BG5 IN	DP2
BG5 OUT	DR2
BG4 IN	DS2
BG4 OUT	DT2

Other pin locations are used in the processor board area and can be used in the future in the memory/option board area. It is recommended that the user contact Cal Data prior to assigning unused pin locations to special functions to ensure future compatibility with all system options.