5-73

PR. 316

LA-5302-MS

INFORMAL REPORT

Programming the MBD Model I

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

MASTER

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

In the interest of prompt distribution, this LAMS report was not edited by the Technical Information staff.

Printed in the United States of America. Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151

Price: Printed Copy ~~\$9.00~~; Microfiche ~~\$4.00~~

\$4.00

\$1.95

The logo features a vertical line with a wavy section, passing through a series of concentric circles. The circles are centered on the line, and the wavy section is located between the top and bottom circles. The text "los alamos" is written in a bold, sans-serif font, with "los" in lowercase and "alamos" in lowercase. Below "los alamos" is the text "scientific laboratory" in a smaller, bold, sans-serif font. Below "scientific laboratory" is the text "of the University of California" in a smaller, sans-serif font. Below "of the University of California" is the text "LOS ALAMOS, NEW MEXICO 87544" in a smaller, sans-serif font.

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

LA-5302-MS

INFORMAL REPORT

UC-32

ISSUED: July 1973

Programming the MBD Model I

by

Richard F. Thomas, Jr.

MASTER

■ DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

A handwritten signature, possibly "Jeg", is written in the bottom right corner of the page.

PROGRAMMING THE MBD MODEL I

by

Richard F. Thomas, Jr.

ABSTRACT

This report describes the LAMPF Microprogrammed Branch Driver (MBD) from a programmer's point of view and describes a set of macro-instructions, implemented in the PDP-11 MACRO assembler, suitable for writing programs for it.

INTRODUCTION

The Microprogrammed Branch Driver (MBD) is a small, special-purpose, stored-program processor intended to serve as an interface between a CAMAC branch and a Digital Equipment Corporation PDP-11. It has 256 words of read-write memory,* 16 bits in length, a simple arithmetic unit, and an elaborate register structure, features which permit it to multiplex a single CAMAC branch among eight data channels to a PDP-11. The instructions, which are generally executed within a 420-nanosecond cycle, permit addition, subtraction, masking, and shifting of 16-bit integers as well as the testing of results, execution of CAMAC commands, and communication with the PDP-11 via the Unibus. Three modes of interaction with the PDP-11 are provided: shared registers which both processor use for communication, read-write access to PDP-11 core memory, and interrupts to the PDP-11 CPU. While the MBD is generally intended to depend on the PDP-11 for supervisory control, program loading, and auxiliary storage, it can, if necessary, be operated independently.

GENERAL ORGANIZATION OF THE MBD

The MBD (Model I) is organized around a large number of programmable registers. Generally, the intention of the design is that all data, whether it is data being transmitted between the PDP-11 and the

CAMAC branch or control data being used for controlling the procedure, is held in registers, and the memory is used to hold instructions and constants. The registers can be classified in several different ways, but Figure 1 shows a classification which is useful in picturing the overall functioning of the MBD. Two data buses form the basic structure of the device. In a normal data-transfer operation, a selected register sends information over the source bus through the arithmetic and logical unit, where it may be modified in various ways, to the destination bus and on to a (usually different) destination register. Most, but not all, registers are connected to both buses and can both send and receive data.

The registers shown in Figure 1 are divided into five classes depending on their functional associations. There is a group of registers which are used primarily for communication with the PDP-11, and these have been designated "Unibus registers." Another group of registers functions primarily to communicate with the CAMAC branch; these registers are shown in the diagram as "CAMAC registers." The group designated "channel registers" generally contains control information (e.g., buffer address, word count, CAMAC command) necessary for performing a particular task. There are eight identical sets of six channel registers, but only one set is in use at any particular instant. The inactive sets contain information, but do not participate in the data transfers. It is by means of these eight sets of channel registers that the branch is multiplexed

*Although the Model I MBD is built with a maximum addressing capability of 1024 words, all existing machines contain only 256 words.

among up to eight different tasks. A priority scheme decides which of the eight sets will be active at any given time. Generally, each set of channel registers is thought of as representing an independent data channel, and hence the name. The fourth group of registers consists of certain miscellaneous registers which are not channel registers and not closely associated with either the Unibus or CAMAC interfaces. They serve various purposes, most typically making available to a program information about the current state of the MBD itself. The fifth class of registers shown in Figure 1 has only one member: the arithmetic transfer register. It receives the result of every data-movement operation and is an implied source of an argument for many instructions. The properties and functions of the registers which are of concern in programming the MBD are described in the sections which follow.

The channel registers are the only ones which are replicated in the MBD. There is only one arithmetic and logical unit, one arithmetic transfer register, one set of Unibus registers, one set of CAMAC registers, and one each of the miscellaneous registers.

The eight channels are ordered by priority, channel 7 having the highest priority and channel 0 the lowest. For a given channel to be selected and to run, at least one of the following three conditions must exist:

1. The PDP-11 must have requested the channel to start a task. Such a request sets a bit called the Channel Initialize Latch (CIL) which is one of the inputs to the priority arbitration logic. There is a CIL bit for each of the eight channels. However, since all CIL bits are reset when an initialize request is accepted, no more than one may be set at any given instant.

2. The Program Request Latch (PRL) must be set. The PRL bit is also an input to the priority arbitration logic, but it can be set or cleared by the MBD program itself. Just as with the CIL bits there is one for each channel.

3. There must be a one in the graded-L bit which corresponds to the channel and the corresponding Channel Enable Latch (CEL) must be set. GL bits 17 to 24 correspond directly to channels 0 to 7, and each channel has a CEL bit which can be set or cleared under program control in the MBD. The input to the priority arbitration logic in this case is the logical product of the GL bit and CEL bit.

The priority arbitration logic actually contains 16 levels rather than 8, with two adjacent levels assigned to each channel, thus permitting each channel to be started in two different ways. For each channel the CIL bit has a higher priority than the PRL bit or the logical product of CEL and GL. If the CIL bit of a given channel is set and no higher priority channel has any request bits set, the channel is selected and started at memory location 1. If the CIL bit for a channel is not set, but its PRL bit or its CEL and GL bits are set, it is selected and started at memory location 0. This two-level-per-channel priority scheme permits the MBD to execute different instruction sequences, depending on the source of the request to run. Furthermore, it permits the PDP-11 to interrupt a program being executed by a channel and force it to execute a different program. In this way, the PDP-11 is able to maintain full control over the activities of the MBD.

Fig. 1

REGISTERS

Channel Registers

Each of the eight channels has six registers, 16 bits in length. Five of the six registers are general, programmable registers in their implementation, although they have been given names which suggest special purposes. One register, the Control and Test Register, in each channel has special properties which restrict its usage for general purposes but provide special facilities for bit testing and transfer of control. The five general purpose registers are:

ILR: Instruction Location Register

DAR: Data Address Register

WCR: Word Count Register

CCR: CAMAC Command Register

GPR: General Purpose Register

These registers are able to receive and supply 16-bit words of data involved in the various two-address MBD instructions. They are generally used to hold the permanent information necessary to permit a channel to resume operation after having yielded control of the branch to another channel, either while waiting for a LAM, or because another channel has a higher priority. The names of the registers suggest their intended usage, but there is nothing in the design of the MBD which requires these suggested usages to be followed.

The special purpose register in each channel (CTR: Control and Test Register) can receive data during a normal two-address, data-movement instruction, but cannot supply it. That is, once it has been written into, the information is not available to be moved from CTR to some other register. The information is available, however, in various special ways. Bits 0 to 7 (the low-order eight bits of the register) are used by a special jump instruction to provide an instruction word address.* The upper six bits (bits 10 to 15) of the CTR can be tested individually by the conditional branch instructions. Bits 8 and 9 are not used. Two modes of loading the CTR are available: The entire register may be loaded if the CTR is specified as the destination in a data-movement instruction, or, the low byte only may be loaded if a different address

is specified in the destination field. In this case, the upper byte is not changed. The low byte can also be loaded by a special immediate address instruction, also without disturbing the high-order bits. To summarize the features of the CTR, it provides a convenient means of testing certain individual bits, and it provides a means in the channel registers for remembering the address of and jumping to a particular instruction in the MBD memory.

Unibus Registers

Three registers addressable by the MBD are used for communication with the PDP-11. All three of these registers can both supply and receive 16-bit words of data in the same way that the channel general registers can, but since they are not replicated, there is no guarantee that the information will remain in them if a different channel should run between storing information in them and fetching it. In addition, each register has a specific, defined use which makes it impractical to use it for other purposes, except for the most fleeting, temporary storage. The names of the registers are:

MDR: Memory Data Register

MAR: Memory Address Register

PDR: Programmed Data Register

The memory data register is used during a transfer to or from the PDP-11 memory to contain the word which is transferred. The memory address register is used during a transfer to or from the PDP-11 memory to contain the address of the word in the PDP-11. Only full words are transferred, and the contents of the MAR is a word address instead of a byte address, as it would be in one of the PDP-11 registers. When the address is presented to the bus, it is shifted left one bit and a low-order zero supplied so that a legal PDP-11 word address appears on the bus. The high-order address bit (bit 17) is also always made zero. Consequently, the MBD is capable of addressing up to 65,536 words of PDP-11 memory, but is not able to control peripheral devices or communicate with them directly. In order to read a word from the PDP-11 memory the address of the word to be read is placed in the MAR and a read command issued. When the Unibus cycle is completed, the contents of the word will be in the MDR. In order to write a word into the

*Note that the MBD uses word addresses while the PDP-11 uses byte addresses.

PDP-11 memory, the information to be written must be placed in MDR and the address placed in MAR before the write command is issued. The order in which the registers are loaded does not matter. MAR and MDR cannot be addressed directly by the PDP-11 processor.

The programmed data register is a communications register accessible to the PDP-11 at address 764002 and also accessible to the MBD via its normal instructions for data movement between registers. The use of this register is defined by programming convention. Normally, it is loaded by the PDP-11 before a channel is started up, and the channel, when it is started, interprets its contents to find out what it is being asked to do. Typically, it indicates which of several programs in the MBD memory to execute, or contains an address in the PDP-11 memory where the MBD may obtain more information about the task. One must be careful not to change the information in the PDR until it has been saved or acted upon by the MBD. The techniques for doing this may involve the setting of flags in core by the MBD, the generation of interrupts, the exchange of codes in the PDR itself,* or the use of the MBD control and status register ready bit. Some convention must be established, however, which permits each processor to determine whether the PDR is currently available for its use.

CAMAC Registers

The programmable registers which are used to communicate with the CAMAC branch are similar in concept and function to those used to transfer data on the Unibus. Their names are:

BAR: Branch Address Register

BDL: Branch Data Register, Low Order

BDH: Branch Data Register, High Order

In order to perform an operation on the CAMAC branch, the branch address register is loaded with a word which contains the FCNA code for the operation (less bit F8). The format of the word is as follows:

Bits 0-3: A (subaddress)

Bits 4-8: N (station number)

Bits 9-11: C (crate number, coded in three bits)

Bits 12-15: F (function code without bit F8)

*Communication through PDR must be done with considerable care, since if the PDP-11 reads the PDR at the same time that the MBD writes it, the PDR will be cleared.

Note that the crate number is coded; in general, multicrate operations are not available. Bit F8 of the function code is supplied by the command which starts the branch operation; there are two such operations, one issuing a function with F8=0, the other issuing a function with F8=1. Consequently, the MBD program must know or have a way of determining the correct value of F8 for each CAMAC command issued. Note that the branch address register is a write-only register; it cannot be read by the MBD, and it must be set before each CAMAC operation, as it is cleared by each operation.

The registers BDL and BDH together form a 24-bit data register capable of receiving or supplying the 24-bit words of the CAMAC system. BDL contains the low-order 16 bits of the word (bits 1 to 16 in CAMAC parlance), and BDH contains the upper eight bits (bits 17 to 24). BDH is only eight bits long; its upper eight bits are not used. BDH and BDL are used on the branch just as MDR is used on the Unibus. They supply the data word for a write instruction and receive the data word for a read instruction. In addition, the 24-bit graded-L word is left in BDH and BDL by a branch graded-L operation.

Miscellaneous Programmable Registers

These registers or pseudo-registers serve special purposes not particularly associated with either the Unibus or CAMAC interfaces, nor are they associated with any particular channel. The first of these is the program counter (PCR), which is eight bits in length. It is used, of course, in the process of stepping through a program in memory. It is available as a source register in normal MBD instructions, but it cannot be used as a destination. It can, however, be stored into by branch or jump instructions. The contents read from PCR by a data-movement instruction are the address of the next instruction in memory after the one which accesses its contents. Timing problems make its use as a source unreliable, however, at addresses containing many consecutive 1's.

A second register is the current channel latch (CCL), which is only three bits in length. It also is available only as a source of data, and it contains the number (from 0 to 7) of the currently active channel. The availability of this piece of information in a program provides a means whereby a program can determine which channel is executing

it, and makes possible, as an example, the use of words in memory to extend the capacity of the channel registers. This can be done by using the channel number as an index into eight-word blocks of memory so that each channel uses a different word in each block and does not reference information stored by other channels.

The register address 0 serves as a source of zeros in any data-movement operation, and as a sink for any data which is stored into it.

The graded-L register (GLR) holds the bit pattern which is read by a BG operation on the branch. The upper eight bits of the register are not addressable by the MBD, but are used to cause the various channels to run on demand from the branch. The lower 16 bits (bits 1 to 16) appear in a register which can serve as a destination in a data-movement instruction but not as a source. The effect of setting a bit in GLR is to generate an interrupt to the PDP-11 if the bit is the leftmost 1 in GLR which has a corresponding 1 in MSK. These sixteen bits are associated with the sixteen interrupt vector locations at byte addresses 400 to 476, and if a bit is set in register MSK (see below in section titled Registers Accessible to the PDP-11), an interrupt at the associated vector location is generated at priority level 5. If GLR has bits set by a GL operation and a program subsequently stores a different pattern into it, while the MBD is in the process of generating the hardware-produced interrupt, the results are unpredictable. If a program stores a bit into GLR and then permits the MBD to execute a GL operation (see exit commands below), the bit set by the software may be cleared, and again results are not predictable. There is no means within the MBD to determine when the PDP-11 has recognized one of these sixteen interrupts; consequently an arrangement of software flags is necessary so that the MBD can find out when it is safe to permit a GL operation to occur.

Arithmetic Transfer Register

The arithmetic transfer register (ATR), a 16-bit register associated with the arithmetic and logic unit of the MBD, is not, strictly speaking, a programmable register in the same sense as the other registers described up to this point. However, it is an implicit operand of many instructions, and

the result of each data-movement operation is left in it. Of all the registers in the MBD it most closely resembles in its functions and usage the classical computer accumulator register.

Registers Accessible to the PDP-11

Four registers are accessible to the PDP-11 via Unibus addresses. One of these, the PDR, is also an internally programmable register of the MBD and has already been described in the section on Unibus Registers. Since the PDR has two different addresses depending on whether it is referenced by the MBD or the PDP-11, it has been given two different names in the documentation and in the assembly macros. When referenced by the MBD, it is called PDR; when referenced by the PDP-11, it is called PDX (external address of programmed data register).

The control and status register (CSR) is the chief means by which the PDP-11 controls the activities of the MBD. It is by setting bits in this register that the MBD is started and stopped, channels are selected, etc. The fields in the word which are written into by the PDP-11 are:

Bit 0: Single-cycle mode
Bits 1-3: Function
Bit 6: Interrupt enable
Bits 8-10: Channel number.

The fields which can be read by the PDP-11 are:

Bit 7: Ready
Bit 13: Branch error
Bit 14: Bus error.

Note that the bits which can be written into by the PDP-11 cannot be read back. The meanings of the various fields are indicated below:

Single-Cycle Mode (Bit 0). If bit 0 of CSR is set to 1, the MBD is put in single-cycle mode. This state is similar to the single-step mode of a computer; it executes only one instruction at a time (on command from the PDP-11) instead of running continuously. This bit can be set only in conjunction with an initialize function (see function bits below). The MBD cannot be switched to a single-cycle mode while it is running.

Function (Bits 1-3). The function field is set by the PDP-11 to certain codes to control MBD operation. In practice, only three of the possible eight codes are of practical use. If the field is set to

1, the MBD is reset, all channels are stopped, the program counter is cleared, and request and enable latches are cleared. If the function field is set to 4, the MBD executes the next instruction, as indicated in its program counter. (This is the means by which it is made to execute a program in the single cycle mode.) If the function field is set to 5, then the channel whose number appears in the channel number field is initialized. That is, a latch (CIL) is set requesting the selected channel to be started by executing the instruction at memory location 1. If at the same time the single-cycle bit is set to 1, the MBD will be in single cycle mode with the channel selected. If the single-cycle bit is 0, instructions are processed automatically without need of further attention from the PDP-11.

Interrupt Enable (Bit 6). If at the time a channel is initialized the interrupt-enable bit is set to 1, the MBD will be able to interrupt the PDP-11. Otherwise, the commands in the MBD which generate interrupts will have no effect; LAM's which are transmitted directly through to the PDP-11 will have no effect, and the error condition will not interrupt. If the MBD is to be capable of interrupting the PDP-11 on a continuous basis effectively, each word stored into CSR must have the interrupt enable bit set to 1. The last command received is the controlling one, and so a new command with the bit set to zero will disable all interrupts.

Channel Number (Bits 8-10). This field is used only in conjunction with the channel initialize function to indicate which of the eight channels is being selected.

Ready (Bit 7). The ready bit indicates that the MBD is ready to accept a new command. Its principal use is to indicate to the PDP-11 that a previously issued channel initialize function has been accepted and that it is safe to issue a new one. When an initialize command is given, it may not be accepted immediately, because the MBD may be busy with activity on a different channel. The ready bit remains a zero until the selected channel actually begins executing the instruction at memory location 1. This point is mainly important because part of the initialization process may involve trans-

mitting information via the PDR, and the ready bit tells the PDP-11 when it is safe to store in the PDR new information intended for a different channel.

Branch Error (Bit 13). In the event of an error the MBD generates an interrupt to the PDP-11 at vector location 540. If the error was a branch timing error, the branch error bit is set.

Unibus Error (Bit 14). If an error interrupt is due to a bus error, bit 14 of CSR is set.

The third register accessible to the PDP-11 is the instruction register of the MBD. If the MBD is in single-cycle mode, any instruction stored into the instruction register is immediately executed. This feature permits relatively easy loading of the MBD memory and simplifies diagnostic and debugging programs.

The fourth and last register in this class is the mask register. The graded-L bits 1-16 are connected to the PDP-11 interrupt vectors 400 to 474 via the register GLR. The mask register provides a means for the PDP-11 to control which interrupts it will accept in this group. If a bit in the mask register is a one, the corresponding interrupt is enabled.

A summary of the MBD addressable registers is given in Table I.

TABLE I
ADDRESSABLE REGISTERS

Name	Adr (octal)	
<u>Channel Registers</u>		
ILR	01	Instruction Location Register
DAR	02	Data Address Register
WCR	03	Word Count Register
CCR	04	CAMAC Command Register
GPR	06	General Purpose Register
CTR	05	Control and Test Register
CTRL	07	CTR Low Byte
<u>Unibus Registers</u>		
MDR	11	Memory Data Register
MAR	12	Memory Address Register
PDR	16	Programmed Data Register
<u>CAMAC Registers</u>		
BAR	15	Branch Address Register
BDL	13	Branch Data, Low Order
BDH	14	Branch Data, High Order

TABLE I Cont'd
ADDRESSABLE REGISTERS

Name	Adr (octal)	
<u>Miscellaneous Programmable Registers</u>		
PCR	07	Program Counter
CCL	17	Current Channel Latch
	00	Source of Zeros
GLR	10	Graded-L Register
<u>Arithmetic Transfer Register</u>		
ATR		Arithmetic Transfer Register
<u>Registers Accessible to the PDP-11</u>		
CSR	764000	Control and Status Register
PDX	764002	Programmed Data Register
IR	764006	Instruction Register
MSK	764004	Interrupt Mask Register

INSTRUCTIONS

Instructions in the MBD are sixteen bits in length and may be in any of seven different formats. Generally, the instruction word is divided into four four-bit fields. The leftmost field, bits 12 to 15, always contains an operation code. The remaining three fields are interpreted in different ways and in some cases combined into larger fields, depending upon the operation code.

Data-Movement Instructions

The format for this class of instructions consists of four four-bit fields as indicated below:

Hexadecimal grouping: 0000 CCCC SSSS DDDD

Octal grouping: 0 000 CCC CSS SSD DDD

"0" indicates bits of the operation code; "C" indicates bits of the control command; "S" indicates bits of the source register address; and "D" indicates bits of the destination register address.

Instructions of this class play a dual role; they generally move information from one register to another, possibly modifying it in the process, and they may command functions, either internal or external, which do not involve internal data movement. Thus, the format allows two operation fields and two register address fields. The first operation field, bits 12 to 15, indicates the data movement function; the second, bits 8 to 11, indicates the independently specified control command. Any control command can be combined with any data-movement operation. The register addresses are those indicated in Table I. Most, but not all, registers can serve as both sources and destinations; see the

section on Registers for details. Several registers are less than sixteen bits in length; unused bits in these registers yield zeros when the contents are accessed as sources. Using an unassigned address code in the source field yields a full word of zeros. The names, numeric codes (in octal), and definitions of these instructions are given below.

MV: 00: Move

Move the contents of the source register to the destination register. The contents of the source register are not changed.

INM: 01: Increment and Move

The contents of the source register is incremented by one and then stored in the destination register and in ATR. If the source register is a channel register, the incremented value also replaces the original contents of the source. This feature makes it possible to maintain in channel registers PDP-11 address pointers which can be updated and moved to MAR in a single instruction.

DEM: 02: Decrement and Move

Decrement the contents of the source register by one and store the result in the destination and in ATR. If the source is a channel register, also replace the original contents of the source by the new result. This instruction is the complement of INM.

AD: 03: Add

Add the contents of the arithmetic transfer register (ATR) to the contents of the source register and store the result in the destination register and in ATR.

SB: 04: Subtract

Subtract the contents of ATR from the contents of the source register and store the result in the destination register and in ATR.

IOR: 05: Inclusive Or

Store the inclusive OR of the contents of ATR and the source in the destination register. Leave the result in ATR.

EOR: 06: Exclusive Or

Store the exclusive OR in the contents of ATR and the source in the destination register. Leave the result in ATR.

AND: 07: And

Store the AND function of the contents of ATR and the source in the destination register. Leave the result in ATR.

The secondary, or control, command is optional. If no control command is required in a given instruction, the control field is set to zero. The control command symbols, numeric codes (in octal), and meanings are as follows.

DCH: 01: Data Channel Read and Write, Holding Unibus

A Unibus read and pause operation is initiated, causing a memory word to be read out but not rewritten. After the read has been completed, the first store into MDR causes the cycle to be completed, and the new contents of MDR is stored back into the word just read. This feature is intended for updating words in PDP-11 core memory in multichannel analyzer applications.

DCR: 02: Data Channel Read

This command initiates a Unibus read of the word whose address is in MAR. (The data-movement operation with which this instruction is associated can be the one which loads MAR.) The bus is reserved, the word is transferred to MDR, and the bus is released.

DCW: 03: Data Channel Write

This command initiates a Unibus write into the word in PDP-11 memory whose address is contained in MAR. The bus is reserved, the word written from MDR, and the bus released. This command is the complement of DCR.

BCO: 05: Branch Command with F8=0

This command initiates a branch operation for which F, C, N, and A are contained (having been previously loaded or being loaded by the data-movement instruction associated with this command) in BAR. The function in BAR lacks the CAMAC function bit F8; this command specifies that F8 for the function equals zero.

BC1: 06: Branch Command with F8=1

This command initiates a branch operation for which F, C, N, and A are contained in BAR. It specifies that the value of F8 for the CAMAC function equals one. This command is the complement of BCO.

EX1: 10: Exit 1

This command causes the channel which executes it to terminate execution, clear the program counter, and permit the channel priority logic to determine whether a higher priority channel requires service. The channel will resume executing instructions at memory location zero whenever its priority is highest if and only if the program request latch has been set for this channel. (See PRL command below.) Whenever any exit command is executed, the branch demand line (BD) is tested automatically by the MBD. If BD=1, the MBD issues a graded-L command (BG), thus reading the GL pattern from the branch. This pattern is loaded into BDH and BDL as well as GLR. If one of the high-order eight GL bits is a 1, it may cause one of the eight channels to begin executing instructions. If BD=0 and there is no request for any channel to run, the MBD goes into an idle state continuously monitoring BD. It will be started either by a LAM or by an initialize command from the PDP-11.

EX2: 11: Exit 2

The Exit 2 command causes the channel which executes it to stop executing instructions, clear the program counter, and permit the priority logic to determine which channel should execute instructions next. This command, in addition, clears the program request latch (PRL) associated with the channel which executes it and sets the channel enable latch (CEL). These operations put the channel in a state such that it will not resume executing instructions until the LAM which is associated with the channel becomes active. Essentially, this is a "go to sleep and wait for LAM" command.

EX3: 12: Exit 3

The Exit 3 command, as do the other two exit commands, causes the channel which executed it to stop execution, clear the program counter and permit the priority logic to determine whether there is another channel ready to run. In addition, it clears the channel enable latch, which must have been set by a previously executed Exit 2 command. The channel will not run again unless the program request latch for this channel has been set or it is restarted by the PDP-11. Normally, this command terminates a program which is LAM-driven, and it is

expected that the channel will be restarted by the PDP-11.

INT: 13: Interrupt PDP-11

The PDP-11 is interrupted at the interrupt vector associated with the currently active channel at level BR5. The MBD program continues executing instructions.

PRL: 14: Set Program Request Latch

The program request latch for the channel which executes this instruction is set. This latch causes the channel to resume executing instructions after executing either an Exit 1 or an Exit 3 command whenever the channel is the highest priority channel requiring service. In order for a channel to run after an exit command, either PRL must be set before the exit, or CEL must be set and the appropriate LAM generated in a module. Otherwise, exit commands function as permanent stops.

CLCEL: 16: Clear Channel Enable Latches

This command clears the channel enable latches (CEL) for all channels. Use with extreme care!

BZ: 17: Generate BZ

Issue a BZ command on the branch. Normally, BZ is issued only when starting up a system, and this command is more likely to be issued in single-cycle mode under control of the PDP-11 than by a channel running autonomously. However, this command is the means by which BZ is issued to the branch, and it can be done by a channel if necessary.*

Branch-on-Condition Instructions

The two branch-on-condition instructions have eight-bit address fields in the following format: Hexadecimal grouping: 0000 CCCC AAAA AAAA Octal grouping: 0 000 CCC CAA AAA AAA. "0" indicates bits of the operation code; "C" indicates bits of the condition code; and "A" indicates bits of the address field. Since the address field contains only eight bits, conditional branches are constrained to operate within a single, 256-word page of the MBD memory. The operations, their numeric codes (in octal) and their descriptions are given below.

*BZ is also issued to the branch by the PDP-11 hardware reset, as when the start switch on the PDP-11 is depressed.

BCT: 10: Branch on Condition True

Branch to the word addressed by the address field if the condition tested by the condition code is true. Otherwise, execute the next instruction in sequence. If the condition field contains zero, this instruction is a no-operation.**

BCF: 11: Branch on Condition False

Branch to the word addressed by the address field if the condition selected by the condition field is false. Otherwise, execute the next instruction in sequence. If the condition field contains zero, always branch.

The conditions which are tested by these two operations indicate the status of Unibus operations, branch operations, the state of certain bits in ATR, and the state of certain bits in CTR. The condition selected by code 0 is always false. The condition names, their numeric codes (in octal), and their meanings are given below:

DCB: 01: Data Channel Busy

This condition indicates whether a previously initiated Unibus operation is still in progress. It is necessary to test and be sure that the Unibus is not busy in order to be sure that data has arrived in the MDR after initiating a data channel read operation. It is also unsafe to initiate any data channel operation unless it is certain that DCB is false. Furthermore, a transfer may be garbled if the registers MAR and MDR are used while the data channel is busy. It is possible, for example, for one channel to issue a write command followed immediately by an exit. If another channel becomes active immediately, the data channel write activity may still be in progress. If the new channel modifies MAR or MDR before the completion of the write, it may be done incorrectly. Consequently, a program must be sure that no transfer is in progress before using MAR or MDR.

** Several no-operation instructions are immediately apparent in the MBD. MOV 0,0 has the side effect of clearing ATR, which may sometimes be a disadvantage. BCT 0,0 leaves ATR intact, but does not permit a control command to be issued. The operations IOR 0,0; EOR 0,0; and AD 0,0 permit a control command to be issued, and leave all the registers intact. The assembly macro CON generates the instruction EOR 0,0 and may have a control command specified in the variable field or not, as the programmer wishes.

BRB: 02: Branch Busy

This condition indicates whether a previously initiated branch operation is still in progress. The same discipline must be observed with respect to the branch as is observed with respect to the data channel. Information read by a read function does not arrive in BDL and BDH until BRB is false,* and it is not safe to initiate a new branch operation until BRB is false. If a channel initiates a branch command and then exits before it is complete, the MBD will not issue BG while the command is in progress even though BD=1. If CIL for a channel is set, then that channel will become active and no graded-L cycle will be executed even though BD=1. Under these circumstances a high-priority LAM could be missed even though an exit command is executed. If no CIL bit is set, the MBD will complete the command and then execute the GL operation.

INB: 03: Interrupt Busy

This condition indicates whether the MBD is attempting to interrupt the PDP-11. One should not attempt to generate an interrupt unless INB is false.

NF: 04: Negative Flag

This condition is true if the high-order bit of ATR contains 1.

ZF: 05: Zero Flag

This condition is true if ATR contains zero, unless a shift (of length greater than zero) intervenes between the instruction loading ATR and the test instruction. A shift instruction sets ZF to false regardless of the contents of ATR.

ZLB: 06: Zero Low Byte

This condition is true if the low byte of ATR contains zero, unless a shift instruction (of length greater than zero) intervenes between the instruction loading ATR and the test instruction. A shift instruction sets ZLB to false regardless of the contents of ATR.

*Any given physical CAMAC branch will have a maximum time for execution of a branch cycle. It may happen therefore that the time to execute a given sequence of instructions exceeds this time and eliminates the need, in some cases, to test BRB.

CF: 07: Carry Flag

This condition is true if the carry bit from the most recent arithmetic operation was a 1 and there have been no intervening register-load operations, store operations, logical operations, or shift operations. Subtraction is done by adding the ones complement of the subtrahend plus 1 to the minuend. If this procedure produces a carry out of bit 15, the carry flag is set. The decrement and move instruction (DEM) always produces a carry except when the source register initially contains zero. The other instructions affect the carry flag as follows: MV, LCI, LDP, STP, and LDT set it to true. IOR sets it false. EOR sets the flag to the same value it would have as if an AD instruction had been executed with the same input values. All shift instructions will set the flag false if the shift count is greater than zero, leave it unchanged otherwise. The AND instruction sets the carry flag to true if the following logical function is a 1, to false otherwise. $CF = R15(T15' + R14(T14' + R13(T13' + R12(T12' + R11(T11' + R10(T10' + R9(T9' + R8(T8' + R7(T7' + R6(T6' + R5(T5' + R4(T4' + R3'(T3' + R2(T2' + R1(T1' + ROT0'))))))))))))$. R_n represents the nth bit from the register input. T_n represents the nth bit from the ATR.

QF: 10: Q Flag

This condition is true if the Q response from the most recently executed branch command was a 1.

XF: 11: X Flag

This condition is true if the X response from the most recently executed branch command was a 1.

C10: 12: CTR Bit 10

This condition is true if CTR bit 10 contains 1.

C11: 13: CTR Bit 11

This condition is true if CTR bit 11 contains 1.

C12: 14: CTR Bit 12

This condition is true if CTR bit 12 contains 1.

C13: 15: CTR Bit 13

This condition is true if CTR bit 13 contains 1.

C14: 16: CTR Bit 14

This condition is true if CTR bit 14 contains 1.

C15: 17: CTR Bit 15

This condition is true if CTR bit 15 contains 1.

Jump via CTR

JVC: 12: Jump via CTR

This single instruction permits a jump or branch command with the branch address held in the right-most bits of CTR; its 16-bit code is

Hexadecimal grouping: 1010 0000 0000 0000

Octal grouping: 1 010 000 000 000 000.

In normal operation, memory location zero always contains a JVC instruction. Before a channel executes an Exit command, it loads CTR with the address of the first word of the next segment of its program. Since a channel normally resumes execution at location zero, the JVC instruction there immediately branches to the correct place to continue the program. And since each channel has its own individual CTR, each channel can be executing from a different part of the program at any given time. This feature is also available for use during ordinary program execution, of course, and provides a convenient subroutine return facility, as an example. Note that the low-order bits of the instruction must be zero for proper execution of this instruction.

Load and Store Instructions

The load and store instructions provide the only means for loading and storing the MBD memory. The format of these instructions is

Hexadecimal grouping: 0000 P000 AAAA AAAA

Octal grouping: 0 000 P00 0AA AAA AAA.

"0" represents operation code bits; "P" is a modifier bit which indicates which of two registers participates in the operation; "0" represents zero bits; and "A" represents address bits. The load and store operate between MBD memory and either ATR or PDR. If the "P" bit in the instruction is 1, PDR participates, otherwise, ATR. Loading and storing PDR is mostly done in single-cycle mode under direct control of the PDP-11 and is a convenient way of loading programs into the MBD. Four mnemonic codes are defined for the two operations and their two variants; the names, octal codes (with the modifier

set off by a period) and meanings are given below:

STT: 13.0: Store ATR

Store the contents of ATR in the MBD memory word selected by the address field.

STP: 13.4: Store PDR

Store the contents of PDR in the MBD memory word selected by the address field and into ATR.

LDT: 14.0: Load ATR

Load ATR from the MBD memory word selected by the address field.

LDP: 14.4: Load PDR

Load PDR and ATR from the MBD memory word selected by the address field.

Shift Instructions

A single shift operation with four modifiers permits the contents of ATR to be shifted up to 15 bits right or left, logical or circular. The format for this set of operations is:

Hexadecimal grouping: 1010 00TD 0000 NNNN

Octal grouping: 1 010 00T D00 00N NNN.

"1" indicates bits which contain 1; "0" indicates bits which contain 0; "T" indicates the type of shift (logical shift or rotate); "D" indicates the direction of the shift (right or left); "N" indicates bits in the shift count. The names given the operations, their numeric codes in octal, with the T and D modifiers set off from the operation code by a period, and the descriptions are given below. "Logical shift" means that zeros are shifted into the register; "rotate" means that bits shifted out of one end of the register are shifted into the other. The carry and zero flags are set to the false condition by all shift operations if the shift count is greater than zero.

SLL: 15.10: Shift Left Logical

The contents of ATR are shifted left logically the number of bits given by the shift count.

SRL: 15.14: Shift Right Logical

The contents of ATR are shifted right logically the number of bits given by the shift count.

SLR: 15.00: Shift Left Rotate

The contents of ATR are rotated left the number of bits given by the shift count.

SRR: 15.04: Shift Right Rotate

The contents of ATR are rotated right the number of bits given by the shift count.

Load CTR Immediate (LCI)

LCI: 16: Load CTR Immediate

This instruction provides a quick and easy way of placing an address constant in the lower bits of CTR. The format is:

Hexadecimal grouping: 1110 0000 AAAA AAAA

Octal grouping: 1 110 000 0AA AAA AAA.

When the instruction is executed, the low-order eight bits of the instruction itself are moved into the low-order byte of CTR, and the result is left in ATR. The carry flag (CF) is cleared.

Jump (JP)

JP: 17: Jump

The jump instruction is a standard unconditional branch instruction, with 12-bit address. The format is:

Hexadecimal grouping: 1111 AAAA AAAA AAAA

Octal grouping: 1 111 AAA AAA AAA AAA.

It has been mentioned that a BCF instruction with a zero condition code functions as an unconditional branch, but it has only an eight-bit address field. The JP must be used to transfer control between 256-word pages and is the recommended instruction for all unconditional jumps.

Table II gives octal representations of the various fields of the MBD instructions shifted to the appropriate positions in a 16-bit word. The table is useful for reading octal dumps of MBD code as well as for writing instructions directly in octal.

ASSEMBLY OF MBD PROGRAMS

Assembly of MBD programs is a relatively simple matter with the PDP-11 macro-assembler.* With this assembler one can assemble MBD code into normal PDP-11 object modules. They are then linked and loaded into the PDP-11 with programs which in turn load them into the MBD. The symbols used in writing MBD code are those used in this report. Each operation is defined by macro definitions available to any program, and all the register

names, control commands, and conditions are pre-defined. Because the MBD uses full-word addressing, while the PDP-11 uses byte addressing, a word location symbol has been defined, and symbols which are to label locations in MBD programs cannot be defined simply by labeling the instruction, as one does in a normal assembler. Macro-11 defines a label as a byte address, which has a value twice as large as the corresponding word address, and is therefore unsuitable for use in the MBD. This word location counter has the symbol "\$" and it always contains the address of the current instruction or, when the processing of an instruction has been completed, the next instruction. Thus, in order to label a word in the MBD program, one might write

SAM = \$

MV ILR, MAR

These two lines of code give the symbol "SAM" a value which is equal to the word address of the MV instruction immediately following. Using this scheme for assembling and loading MBD programs yields absolute assemblies only. There is no facility for generating or loading relocatable object code.

The conventions which have been adopted with regard to the order of the fields in MBD instructions are:

1. The principal MBD operation appears in the normal instruction field. This is followed by a variable field (as in PDP-11 instructions) which specifies any remaining parameters of the instruction.
2. In data-movement instructions, the subfields are: source register, destination register, control command.
3. The control command field can be left blank, in which case a control field with value zero (no operation) is assembled.
4. In branch-on-condition instructions the branch address precedes the condition code.

The MBD instructions and their fields in assembly format are listed below. Subfields of the variable field which are optional are enclosed in square brackets.

* See Macro-11 Assembly Programmer's Manual, DEC-11-OMACA-A-D, Digital Equipment Corporation, Maynard, Massachusetts.

TABLE II
OCTAL REPRESENTATION OF INSTRUCTIONS

<u>Operation</u>	<u>Control</u>	<u>Condition</u>	<u>Source</u>	<u>Destination</u>
000000 MV	0000	0000	000	00
010000 INM	0400 DCH	0400 DCB	020 ILR	01 ILR
020000 DEM	1000 DCR	1000 BRB	040 DAR	02 DAR
030000 AD	1400 DCW	1400 INB	060 WCR	03 WCR
040000 SB	2000	2000 NF	100 CCR	04 CCR
050000 IOR	2400 BCO	2400 ZF	120	05 CTR
060000 EOR	3000 BC1	3000 ZLB	140 GPR	06 GPR
070000 AND	3400	3400 CF	160 PCR	07 CTRL
100000 BCT	4000 EX1	4000 QF	200	10 GLR
110000 BCF	4400 EX2	4400 XF	220 MDR	11 MDR
120000 JVC	5000 EX3	5000 C10	240 MAR	12 MAR
130000 STT	5400 INT	5400 C11	260 BDL	13 BDL
134000 STP	6000 PRL	6000 C12	300 BDH	14 BDH
140000 LDT	6400	6400 C13	320	15 BAR
144000 LDP	7000 CLCEL	7000 C14	340 PDR	16 PDR
150000 SLR	7400 BZ	7400 C15	360 CCL	17
150400 SRR				
151000 SLL				
151400 SRL				
160000 LCI				
170000 JP				

MV src,dst[,ctrl]
INM src,dst[,ctrl]
DEM src,dst[,ctrl]
AD src,dst[,ctrl]
SB src,dst[,ctrl]
IOR src,dst[,ctrl]
EOR src,dst[,ctrl]
AND src,dst[,ctrl]
BCT adr,cond
BCF adr,cond
JVC
STT dst
STP dst
LDT src
LDP src
SLL cnt
SRL cnt
SLR cnt
SRR cnt
LCI adr
JP adr

CON [ctrl]
DEP dst[,ctrl]

Two pseudo-instructions, defined for convenience in programming are listed above. The operation CON is defined as EOR 0,0[,ctrl]; it is a data-movement operation which does not change any registers and is intended to be used when a control command is to be issued without any associated data-movement. The operation DEP (deposit contents of ATR in destination) is defined as EOR 0,dst[,ctrl]; it is one of the several operations which will move a word from ATR to a normal destination register.

Five data reservation and definition macros have been defined so that the value of the word location counter, \$, can be maintained through such definitions.

Space Reservation:

RSWD exp

This macro reserves the number of words specified in the variable field and sets both \$ and

(the PDP-11 location counter) appropriately.

Data Definition:

OCT exp

The macro OCT stores one word of data, equal to the value of the expression. \$ and . are updated appropriately. If the expression contains numeric fields, they are taken as octal numbers.

DCM exp

The macro DCM stores one word of data, equal to the value of the expression. \$ and . are updated appropriately. If the expression contains numeric fields, they are taken as decimal numbers.

DATA exp1,exp2,exp3,...

The macro DATA stores one word of data for each expression in its variable field up to a maximum of 20. The normal radix conventions of MACRO-11 apply to numeric fields. \$ and . are updated appropriately.

FCNA exp1,exp2,exp3,exp4

The macro FCNA combines its four fields into a single FCNA word, as required for the MBD. The first field gives the function code, the second the crate number, the third the station number, and the last the subaddress. The full function code is written in the first field, and bit F8 is eliminated by the macro. If the fields contain numeric constants, they are taken as decimal.

To use the assembly facilities described above, it is necessary to have the LAMPF version of the system macro file, SYSMAC.SML, available to MACRO-11. Then it is necessary only to use the macro call facility to call one macro and then invoke it in order to define all the symbols and special facilities needed for an MBD assembly. Thus,

.MCALL BDINST

BDINST

These two lines of code are required to prepare MACRO-11 for assembling MBD code.

PROGRAMMING CONSIDERATIONS

Since the MBD is a very general device, relatively little can be said about how it should be programmed. Certain features of its design, however, require a few rules to be followed. Whenever a channel is initialized by the PDP-11, it begins executing instructions at location 1. Consequently, the code which begins at location 1 will normally be common

to all channels and will, at a minimum, have to determine which of several different programs the channel is to execute. When a channel begins execution after having executed an exit command, it begins at location 0. Location 0 always contains a JVC instruction; whenever a channel executes an exit command from which it expects to return, it loads its CTR with the address of the instruction at which it wishes to begin at its next turn. When the priority structure permits the channel to run again, it will begin with the JVC at location zero then branch immediately to the desired location in the channel program.

A channel is started by moving a word containing the initialize function (function 5), the channel number, and if desired, the enable-interrupt bit to CSR. Since all channels begin executing code at the same place, memory location 1, the channel must have a means of distinguishing various tasks. An easy way to do this is to have previously loaded PDX with a one-word message which will serve to guide its future activities. In order for real-time tasks to proceed, a program should execute an exit command fairly frequently. When any exit is executed, the channel stops momentarily and the BD signal on the branch is tested. If BD is a one, the MBD automatically issues BG to the branch and reads the graded-L word into BDH, BDL, and GLR. The channel priority logic then selects the highest priority channel which has a request to run and starts it executing instructions at memory location zero. If a channel program does not contain exit commands, there is no way that any other channel can run, no matter what its priority may be; there is no interrupt facility in the usual sense. Before a channel issues an exit command, it must be sure that all the information it requires to continue execution is in its channel registers, as only these are guaranteed not to be changed before it regains control. Also, if it is not to be restarted by a LAM, it must set PRL sometime during its execution before executing the exit.

The correct operation of the MBD in the modes described above depends on the program counter (PCR) containing zero when a channel begins execution. In normal operation there will be no trouble with this, as PCR is cleared by a reset or by any exit command. However, it is possible to start the MBD

without resetting it after executing instructions in the single-cycle mode, and under these circumstances PCR may not be cleared. In the single-cycle mode the program counter is incremented as each instruction is executed. When a channel is started due to its having its channel initialize latch (CIL) on, the program counter is incremented by one and the instruction at that address is fetched. Consequently if the program counter does not contain zero at that time, the wrong program will be executed. If an application requires switching the MBD from single-cycle to run mode without resetting, an instruction to clear the program counter should be executed just before switching to the run mode.

A channel normally interrupts the PDP-11 to notify it of the completion of a task. However, the interrupt facility is quite general; an interrupt may be issued at any time for any purpose. A channel can interrupt at its dedicated vector location

by executing the INT control command. It can also interrupt at any of the sixteen lower GL positions by storing into GLR. Note also, that the MBD can, if it wishes, modify the contents of interrupt vectors so as to make a single vector point, at different times, to different interrupt servicers. All MBD interrupts are at level BR5.

SAMPLE INITIALIZE PROGRAM

The program shown on p. 16 is given to illustrate the MBD assembly language and to illustrate the kind of process which must necessarily reside in the first few words of the MBD memory. The program starting at location 1 loads ILR from PDR, assuming that the PDP-11 has previously loaded PDR with the byte address of a block in core where will be found values for loading CTR, DAR, WCR, and CCR. It then fetches the first four words out of that block and loads the registers before executing an Exit 1 command.

TABLE III
INTERRUPT VECTORS

400	GL Bit	1
404	GL Bit	2
410	GL Bit	3
414	GL Bit	4
420	GL Bit	5
424	GL Bit	6
430	GL Bit	7
434	GL Bit	8
440	GL Bit	9
444	GL Bit	10
450	GL Bit	11
454	GL Bit	12
460	GL Bit	13
464	GL Bit	14
470	GL Bit	15
474	GL Bit	16
500	Channel	0
504	Channel	1
510	Channel	2
514	Channel	3
520	Channel	4
525	Channel	5
530	Channel	6
534	Channel	7
540	Error Interrupt	

LOC	OPN	VAR FLD	
Ø	JVC		;INITIAL JVC
1	MV	PDR,Ø	;PUT CONTENTS OF PDR IN ATR.
2	SRL	1	;CHANGE BYTE ADDRESS TO WORD
			ADDRESS
3	DEP	ILR	;STORE ATR IN ILR.
4	MV	ILR,MAR,DCR	;MOVE ADDRESS
			TO MAR AND INITIATE DATA CHANNEL
			READ
5	BCT	\$,DCB	;WAIT UNTIL READ PROCESS FINISHES
6	MV	MDR,CTR	;STORE WORD WHICH WAS READ
			INTO CTR
7	INM	ILR,MAR,DCR	;READ NEXT WORD
1Ø	BCT	\$,DCB	;WAIT FOR COMPLETION
11	MV	MDR,Ø,PRL	;MOVE BYTE ADDRESS TO ATR.
			SET PRL
12	SRL	1	;CONVERT TO WORD ADDRESS
13	DEP	DAR	;MOVE ATR CONTENTS TO DAR
14	INM	ILR,MAR,DCR	;READ NEXT WORD
15	BCT	\$,DCB	;WAIT FOR COMPLETION
16	MV	MDR,WCR	;MOVE TO WCR
17	INM	ILR,MAR,DCR	;READ NEXT WORD
2Ø	BCT	\$,DCB	;WAIT FOR COMPLETION
21	MV	MDR,Ø	;PUT WORD IN ATR
22	SRL	1	;CONVERT TO WORD ADDRESS
23	DEP	CCR,EX1	;PLACE IN CCR AND EXIT

INDEX

AD	7	C10	10
AND	8	C11	10
Arithmetic Transfer Register	5	C12	10
Assembly of Programs	12	C13	10
ATR	5	C14	11
		C15	11
BAR	4	CAMAC Branch Operations	4
BC0	8	CAMAC Registers	4
BC1	8	CCL	4
BCF	9	CCR	3
BCT	9	CEL	2
BDH	4	CF	10
BDL	4	Channel Initialize	6
Branch Busy	10	Channel Number	6
Branch-on-Condition	9	Channel Priority	4
BRB	10	Channel Registers	3, 6
BZ	9	CIL	2
		CLCEL	9
		CON	13

Control Commands	8	LCI	12
CSR	5	LDP	11
CSR Function Bits	5	LDT	11
CTR	3	Load	12
DAR	3	MAR	3
DATA	14	Mask Register	5
Data Channel Busy	9	MDR	3
Data Definition	13	MSK	5
Data-Movement Instructions	7	MV.	7
DCB	9	NF.	10
DCH	8	OCT	14
DCM	14	Octal Representation of Instructions.	13
DCR	8	PCR	4
DCW	8	PDR	3, 5
Definition of MBD Macros	13	PDX	5
DEM	7	Priority	4
DEP	13	Priority Arbitration Logic	2
EOR	7	PRL	2, 9
EX1	8	QF.	10
EX2	8	Ready Bit	6
EX3	8	Register 0.	3
FCNA	4	Register Addresses	3
FCNA Format.	4	Register Classes	3
General Organization	1	Registers	3
General Purpose Registers	3	Registers on Unibus	3
GLR	5	Reserving Memory.	13
GPR	3	RSWD.	13
ILR	3	SB.	7
INB	10	Shift	11
INM	7	Single-Cycle Mode	5
Instruction Formats	12	SLL	11
Instruction Register	4	SLR	11
Instructions	7	SRL	11
INT	9	SRR	12
Interrupt Enable	6	Starting a Channel	2
Interrupt Level	9	Store	11
Interrupt Vectors	15	STP	11
Interrupts	9	STT	11
IOR	7	Table I	6, 7
IR	7	Table II	13
JP	12	Table III	15
Jump	12		
JVC	11		

Unibus Registers	3
Unibus Transfers	3
WCR	3
XF	10
ZF	10
ZLB	10