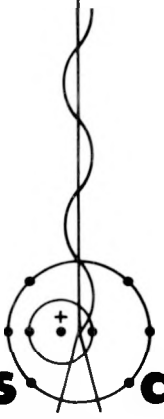


223
~~5-30~~

127

LA-5144

Design and Operation of a
Microprogrammed Branch Driver
for a PDP-11 Computer



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



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

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of 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. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

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.

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 \$3.00; Microfiche \$0.95

LA-5144

UC-32 and 28

ISSUED: May 1973



Design and Operation of a Microprogrammed Branch Driver for a PDP-11 Computer

by

Lavon R. Biswell
Robert E. Rajala

NOTICE

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.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

flg

DESIGN AND OPERATION OF A MICROPROGRAMMED BRANCH DRIVER FOR A PDP-11 COMPUTER

by

Lavon R. Biswell and Robert E. Rajala

ABSTRACT

A microprogrammed branch driver (MBD) is the interface between PDP-11 series computers and a multirate CAMAC system. It is a microprocessor-controlled, multiple, direct-memory-access (DMA) channel branch driver.

The MBD requirements stem from design of the Clinton P. Anderson Meson Physics Facility data-acquisition system. The desire for a standard system, the types of experiments and data rates, and the varying complexity of the data-acquisition systems dictated a microprogrammable, multiple DMA channel branch driver. A brief description of CAMAC aids in understanding MBD operation and design.

Trade-offs between cost and capabilities, and decisions that led to the final design are discussed. Computer and CAMAC interface requirements dictated much of the design; the rest was determined by the desirability of a microprocessor with a read-write control memory.

To facilitate checkout, and to serve as a manual controller when the MBD was used as a stand-alone branch driver, a PDP-11 simulator was built. A control-panel option simplified loading of the bootstrap loader for a stand-alone system. After the bootstrap was loaded, the control programs could be loaded from a teletype, paper-tape reader, or data link.

Basic MBD operation and capabilities are discussed. Programs to load the MBD memory from the PDP-11, to initialize the channels, and to control programs are given, and performance specifications are summarized.

I. INTRODUCTION

The Clinton P. Anderson Meson Physics Facility (LAMPF) was built to produce beams of pi and mu mesons 1,000 to 10,000 times more intense than any presently available. The meson facility will also provide intense beams of protons, neutrons, gamma rays, and neutrinos, all of which will be used to study the fundamental properties of nuclear matter.

The heart of the facility is a linear accelerator designed to produce a proton beam with energy variable up to 800 MeV and average current variable up to 1mA. This proton beam is directed onto targets, typically of carbon or tungsten, to produce the meson beams. The accelerator

will pulse 120 times per second with a pulse length of 500 μ sec, giving a 6% duty factor.

In August 1970, a group met at Los Alamos to develop a system design for the LAMPF on-line data-acquisition facilities. The results of this study are documented in the report, "LAMPF Data-Acquisition System."¹ We summarize that report briefly to establish the requirements for the microprogrammed branch driver (MBD) discussed here.

It was concluded that the system should be developed around a small, dedicated computer. The computer would acquire, preprocess, and record data; monitor the conditions of the experimental apparatus; execute necessary control functions; and perform preliminary analyses to

forecast the results of the experiment. It was proposed that LAMPF provide hardware and software support for the standard system. Because of the need to share expensive peripherals and to have access to the arithmetic capability of a large computer, it was recommended that a computer-based terminal be installed at LAMPF and linked to the Central Computing Facility (CCF) at the Los Alamos Scientific Laboratory (LASL). All of the dedicated computers would have a link to the terminal, which would give access to its own peripherals, accelerator data, and a path to the CCF.

It was recommended that, to accommodate a large number of LAMPF users with varied interests, all interfacing between the experimental and dedicated computers be implemented in the CAMAC standard.^{2,3} The study led to the selection of the Digital Equipment Corporation's (DEC) PDP-11 series of computers^{4,5} as the standard for the LAMPF data-acquisition systems and the terminal computer.

The CAMAC system of instrumentation has been adopted by many laboratories in Europe and the United States as a standard method of interfacing research apparatus to establish a stable boundary between instrumentation and computing. A brief description of CAMAC will help clarify the objective of the unit described here. CAMAC replaces the great variety of I/O buses found on computers with a single, nonproprietary design, standardized both mechanically and electrically. The system features a "crate" that will accept up to 24 modules and a

branch that will accept up to seven crates. The crate dataway and the branch highway provide the communication link to the computer. The CAMAC specification restricts the instrumentation contained in a module only to the extent necessary to ensure compatibility with the crate and dataway.

The crate dataway has a 24-bit read bus, a 24-bit write bus, and a control bus. The maximum transfer rate on the dataway is one 24-bit word per microsecond. The control bus provides for 16 subaddresses and for performing up to 32 different operations on a module (function codes). Provision is made for polling the stations via a common "Q" response line and for verifying valid commands via a common "X" response line. In addition to the buses, each individual module has a pair of private lines (N-lines) for module-station selection and one LAM (look-at-me), for its service requests. Modules communicate with the crate controller via a passive multiconductor dataway. Eighty-six pin-edge card connectors allow individual modules to be plugged into the dataway.

Table I is a summary of CAMAC commands and responses. Each module is addressed by means of a single line from the crate controller position; therefore, addressing depends upon module location. If a module is relocated, the computer program must take this into account.

In the same fashion, a module may initiate an interrupt by means of a direct line from each module position to the controller. Separate 24-bit read and write data buses

TABLE I
FUNCTIONS WITHIN CAMAC CRATE

Symbol	Operation	Module	CAMAC Crate	Use
N	Station Number	←	→	Crate controller can select individual Modules No. 1 to 24.
L	Look At Me	→	←	Each module can interrupt crate controller.
A	Subaddress	←	→	Allows 16 subdivisions (e.g., scalars) of each module.
F	Function	←	→	32 different function codes available.
R	24 Read Lines	→	←	Data from module to controller.
W	24 Write Lines	←	→	Data from controller to module.
B	Busy	←	→	Busy.
I	Inhibit	←	→	Inhibits any activity as desired.
C	Clear	←	→	Clears flip-flops, registers, etc.
S	Strobe	←	→	Two timing strobes for data transfer.
Q	Response	→	←	Are you there?
Z	Initialize	←	→	Sets up initial conditions.

Module → CAMAC crate indicates communication from modules to crate controller.
 Module ← CAMAC crate indicates communication from crate controller to modules.

have been specified. The use of separate read and write dataways allows use of less expensive module line drivers than would be required for a combined read-write dataway.

Originally it was envisioned that single-crate systems would interface to small computers, through a computer controller. Therefore, it would be necessary to have one controller available for each crate as well as each type of computer. It was soon apparent that many single-crate systems would expand into those involving more than one crate. Also a good many of the controller functions for a PDP-11, for example, were common to controllers for all computers.

The branch-highway concept described in Report EUR-4600e³ provides a viable solution to all of the above problems. Figure 1 shows how up to seven crates may be

interfaced through a branch driver to single computer. The physical length of the branch highway is limited, the upper limit depending on signal-amplitude loss from line drops and on the delay time one is willing to allow for the appropriate response after a branch-highway data operation has taken place.

The branch highway has two methods of operation. In the command mode, it operates as an extension of the dataway outside the crate. That is, most of the lines in the branch then perform functions similar to those in the crate. However, when not in the command mode, these same lines are available for interrupt requests. Interrupt (look-at-me) signals from any part of the branch typically request that a particular sequence of commands take place. The demand-handling features of the branch highway provide a means for communicating service requests through the branch driver to the computer. With 24 lines available, one can use up to 24 different interrupt requests. This second mode is called either the graded look-at-me, or the graded-L, mode. In addition to this multilevel interrupt feature, a single branch demand line indicates the presence of one or more demands on the 24 graded-L lines. Information transfers in either the command mode or the graded-L mode are appropriately interlocked and take transmission delays into account.

The individual module and crate functions enumerated in Table I are sent through the branch driver to and from the computer. The 24 read and 24 write dataway lines in the crate have been combined into one set of 24 read-write branch lines. In the graded-L mode, they also double as 24 interrupt lines as described above.

Seven crate addresses have been specified, with one line going to each crate. Timing of all transfers is controlled by branch transfer signals. These allow transfers involving both single- and multiple-crate operations. In multiple-crate operations, it is imperative that the branch driver wait for a response from the remotest crate before continuing data transfer.

The above characteristics are not embodied in the I/O bus of any computer; therefore, CAMAC must itself be interfaced to the computer. The interface must resolve the line and logical differences between the two structures, and its design will significantly influence the performance of the total data-acquisition system.

To summarize: The unit must interface the PDP-11 computer to the CAMAC branch highway and be able to operate at the highest possible transfer rate between the two asynchronous systems. It is desirable to free the PDP-11 host computer for as much real-time data analysis as possible. Many of the experiments will have very high event and data rates.

To minimize the PDP-11 load, it is desirable to make direct-memory-access (DMA) transfers rather than

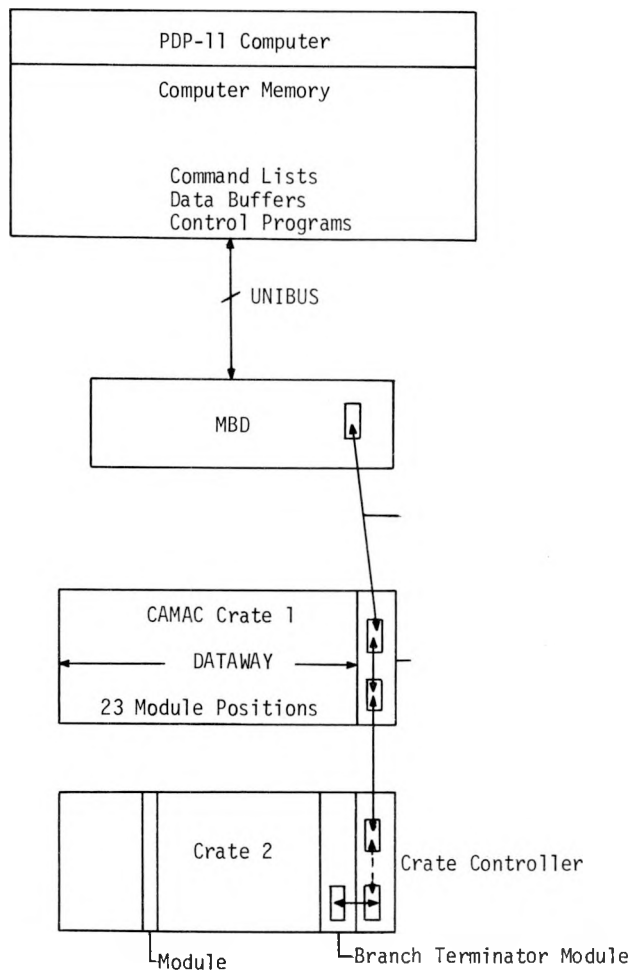


Fig. 1.
LAMPF data-acquisition system.

programmed I/O transfers. The minimum system requirements are two DMA channels for experimental data (scalers, wire chambers, pulse-height analyzers, etc.), one to display accumulated data on a storage scope and the other for communication with the LAMPF terminal through the CAMAC data link.

Secondary requirements are that the unit operate as a stand-alone branch driver and be capable of code self-loading through a control panel or a CAMAC interface to a paper-tape reader or teletype, and that the unit operate as a remote branch driver using the LAMPF high-speed data link.

Because of these requirements, it was decided that a multiple DMA channel branch driver that could be modified or programmed easily was needed. With the advent of microprogrammable processors, this would give the required flexibility and exploit the maximum capabilities of the PDP-11 computer and the CAMAC system. Therefore, this approach was taken in the MBD design discussed here.

II. MICROPROGRAMMED BRANCH DRIVER DESIGN

A. General Design and Approach

The design approach taken for the MBD was not the formal Boolean expression-equation design, but more a systematic trial and error approach with definite goals and limitations. The goals or requirements have been defined, and the limitations are primarily cost and technology. Within the range defined, if design freedom did exist, the approach was not to cut the cost as much as possible but to make the device as flexible and as general-purpose as possible. The primary requirement for the MBD is to interface a PDP-11 computer to a CAMAC branch highway and to operate with the highest possible transfer rate between the two systems.

Several commercial branch drivers are available. The programmed I/O types have typical 20- to 30- μ sec/word transfer rates. A few have one DMA channel with 5- to 10- μ sec/word rates on the channel. The units cost from a thousand to \$6,500, and one with two DMA channels sells for over \$10,000. None of these units has the speed or flexibility required by the LAMPF system. A somewhat arbitrary cost range of \$6,500 to \$10,000 was established for the MBD. The range was from the commercial cost of a branch driver to, at the upper end, the cost of a minicomputer.

The first phase of the design was familiarization with the PDP-11 computer interface and system operation and definition of the requirements for the LAMPF data-acquisition system. The next major decision was to

determine what type of branch driver to build. The choice was between hardware-controlled and microprocessor-controlled multiple DMA channel branch drivers. After study of several articles⁶⁻⁹ on the advantages of microprogramming and some hardware cost analyses, it was decided that the microprocessor-controlled BD was the least expensive, most flexible approach without any overall loss in transfer speeds. With this selection made, the design could be broken down into three major areas, the PDP-11 computer interface, the CAMAC branch driver, and the microprocessor-controlled DMA channel multiplexer, as shown in Fig. 2. Figure 3 shows the MBD and interface cables.

The PDP-11 interface and CAMAC branch driver designs were most heavily influenced by the respective requirements of each system. The rest of the design became a complicated series of trade-offs based on requirements, costs, logic and hardware selections, timing, modes of operation, operating flow diagrams, and arbitrary decisions. Design sources and the major factors in design of each section will be discussed in detail, whether original, from previous designs, from interface manuals, or from application notes. The order in which items are discussed does not reflect the order of the design because, after a couple of passes through the design or redesign, there is no order. However, the general order and the major decisions will be apparent. The decisions about the types of logic and hardware to be used were influenced by familiarity with, and confidence in, the hardware, as well as system speed, power requirements, and cost.

B. Modes of Operation

Table II is a list of MBD operation modes.

Basic MBD operation is, first, to obtain a CAMAC command (17 bits) from a list in the PDP-11 computer memory by a DMA channel operation. The MBD then issues the command to the CAMAC branch and waits for the return of the data word (24 bits). The MBD then transfers the data to a data list in the PDP-11 memory via channel operations controlled by programs stored in the MBD control memory. This programmed mode of operation is the *link-list* mode. Each MBD channel can be operating in a separate link-list mode with the operations interleaved according to channel priority and source of the request.

The MBD when operating as a channel multiplexer and executing instructions from control memory is in the *run mode*. When in the run mode, the MBD cannot be interrupted but must execute a halt or pause to look for higher priority requests. The MBD is placed in run or single-cycle mode by the control register from the PDP-11.

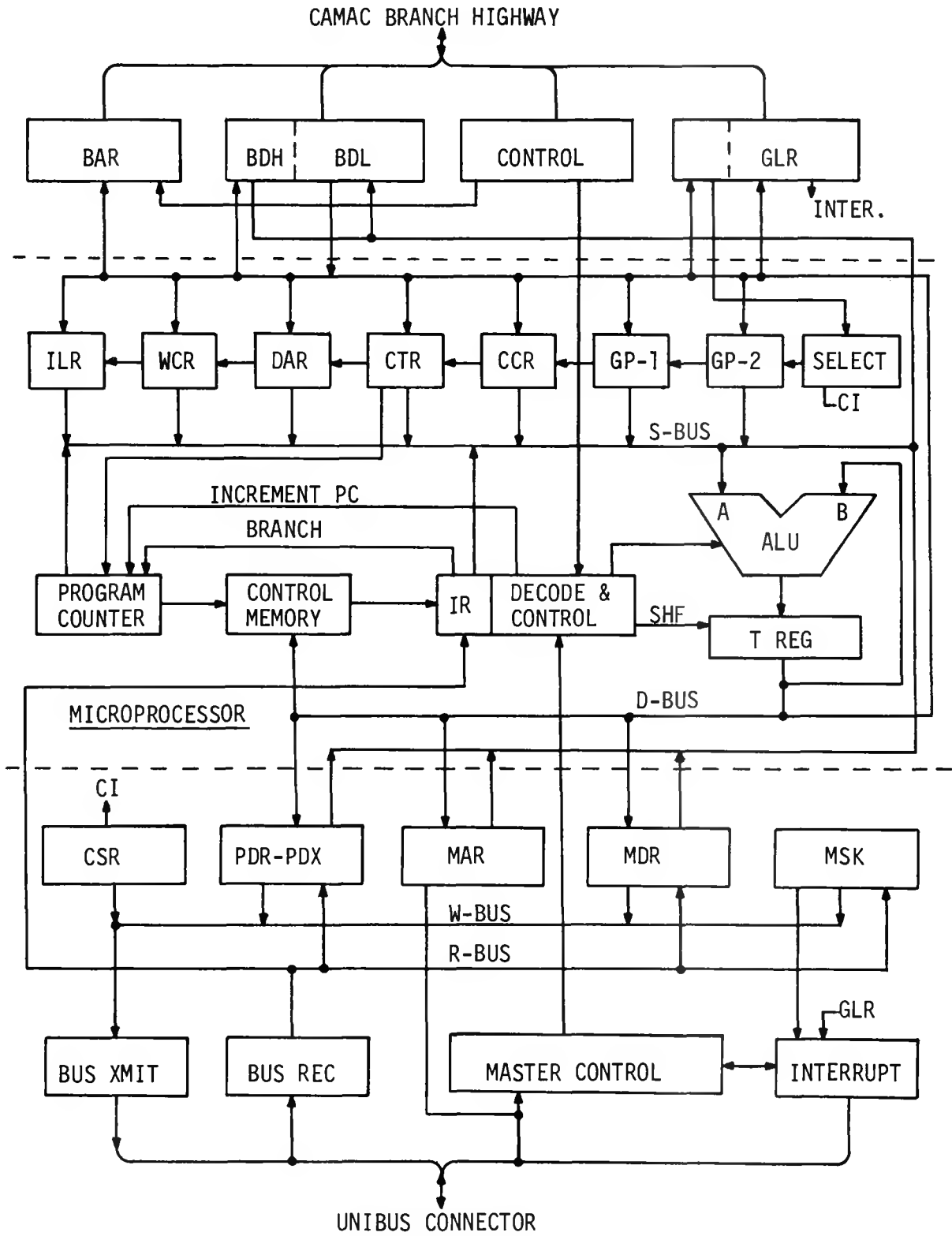


Fig. 2.
Microprogrammed branch driver block diagram.

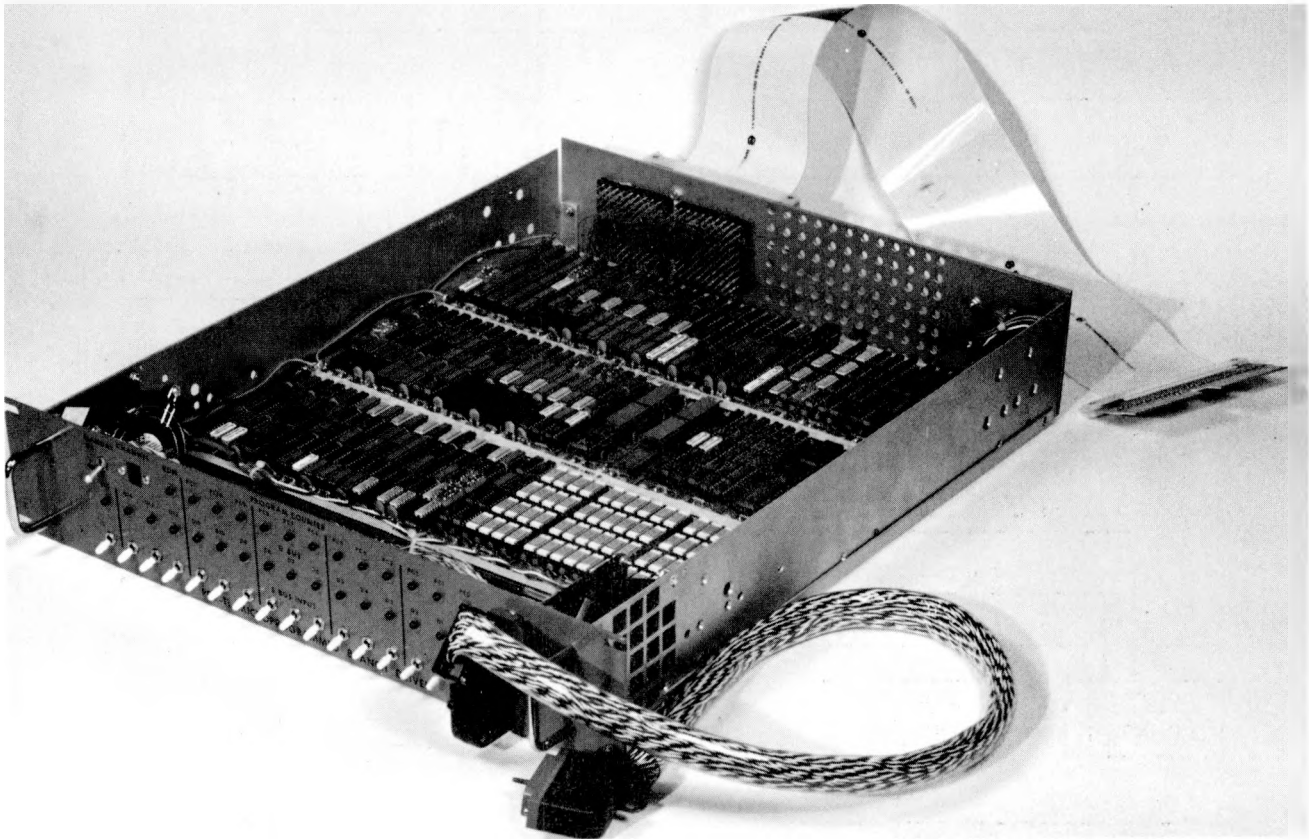


Fig. 3.
Microprogrammed branch driver.

TABLE II
MODES OF OPERATION OF MBD

Mode	Type	Purpose or Operations
Link-List	Programmed	CAMAC commands and data list in PDP-11 memory
Processor	Programmed	CAMAC command in MBD and data list in PDP-11
Block	Programmed	CAMAC command and data in MBD memory
Run	Hardware	MBD operating as channel multiplexer
Single-Cycle	Hardware	Test mode, MBD under control of PDP-11
Manual	Hardware	Load MBD control memory from panel switches

The *single cycle* mode is used to load the control memory to check out and test the MBD. In this mode, the instruction to be executed is loaded into the instruction register (IR) from either the PDP-11 or the manual controller. After being loaded, the instruction is automatically executed and the MBD halts. The manual controller can transfer instructions at a 100-KHz rate that makes testing with an oscilloscope very easy.

The MBD has a *manual mode*, selectable by a front-panel switch, that allows transferring a 16-bit word to the "S" bus from switches on the front panel of the MBD. This mode is used for loading control programs (bootstraps) into the MBD when it is being used as a stand-alone branch driver. The bootstrap allows one to load the rest of the control programs from an input device in the CAMAC branch. CAMAC interfaces are available for teletypes, paper-tape readers and punches, and magnet-tape recorders.

Many modes of program operation can be conceived; a couple that are useful for high-speed operation are the *processor mode* and the *block-transfer mode*. In the

processor mode, the CAMAC commands are stored in the control memory and one DMA transfer is saved, per operation, over the link-list mode. In the block-transfer mode, the data are stored in the control memory then block transferred to the PDP-11. This allows fast operation of the branch; but there are problems with the PDP-11 computer system if the transfer times exceed the service times required by other high-speed devices on the computer, primarily the disk.

The MBD has many modes of operation that allow the user to select the mode and control the memory size required for his optimum operation.

C. PDP-11 Computer Interface

The MBD's computer interface is a standard DEC interface that uses transistor-transistor logic (TTL).

The PDP-11 series of computers has found wide acceptance for real-time data acquisition, analysis, and control. The PDP-11 has one high-speed bus, the Unibus, that connects processor, memory, and all peripherals. The Unibus enables the processor to view peripheral devices as active memory locations. Therefore, memory reference instructions can operate directly on control, status, or data registers in peripheral devices. The PDP-11 has a unique combination of powerful features not previously found in 16-bit computers.

A four-level automatic priority interrupt system permits the PDP-11 central processor (CPU) to respond automatically to conditions outside the CPU. Each peripheral device has a hardware pointer to its own unique pair of memory words, one of which points to the devices' service routine. This eliminates the need for polling devices to identify an interrupt.

The devices' interrupt and service routine priorities are independent. This allows dynamic adjustment of system behavior in response to real-time conditions. The interrupt priority system allows the processor to compare its own priority with that of any interrupting device and to acknowledge the higher level. Interrupt servicing for a device can be interrupted to service a higher priority device. This is nested interrupt servicing, and it can be carried to any level.

The interrupt-handling and subroutine-call hardware are designed to facilitate reentrant code that allows subprograms to be shared by more than one process. This reduces the amount of core required for multi-task applications. The PDP-11 has eight general-purpose registers that can be used as accumulator, pointer to memory, or full-word index registers. Registers 7 and 8 are used as stack pointer and program counter, respectively. An important feature of the instruction set is the availability

of double-operand instructions. These allow memory-to-memory processing and eliminate the need for temporary storage registers. Much of the PDP-11's power is derived from its wide range of addressing modes, which include list sequential, full address indexing, full 16-bit word addressing, 8-bit byte addressing, stack addressing, and direct addressing to 32-K words.

The PDP-11's memory and processor operations are asynchronous. Communications between devices on the bus are in the form of a master-slave relationship. The controlling device is the bus master, the other is the slave. The transfer on the bus is interlocked for each control issued by the master; then there must be a response from the slave. The maximum Unibus transfer rate is one 16-bit word every 750 nsec. When a device can become bus master and request use of the bus, it is generally for one of two purposes: (1) to make a nonprocessor transfer of data directly to memory or (2) to interrupt program execution and force a branch to an interrupt service routine.

The direct memory access (DMA) channel, used by the MBD in the run mode, is a nonprocessor request (NPR) to the PDP-11 computer. An NPR request has the highest priority of any bus request and therefore provides very fast access to the bus. An NPR device can gain bus control in 3.5 μ sec and can make subsequent transfers at memory speed, 1.2 μ sec per 16-bit word. The MBD is bus master during all channel or NPR operations from the MBD. The two MBD registers that control the channel operations are the memory data register (MDR) and the memory address register (MAR).

In addition to these channel registers, the MBD has four registers that are addressable from the PDP-11 and are used for control, initialization, and testing. The control and status register (CSR) is used for setting mode (run or single), initializing the channel, and identifying types of errors. The program data register (PDX) is used as a general-purpose data register in single-cycle mode and as an address pointer in run mode. The mask (MSK) register is an enable for CAMAC interrupts. The MBD instruction register (IR) is writable only in the single-cycle mode. When the IR is loaded, the instruction is automatically executed and the MBD is halted. This mode is used to test and load the control memory. A programmable plug allows any DEC-authorized address to be assigned to the four registers. The registers are buffered, and appear to the Unibus as a single-unit load. The DMA channel looks like one channel to the Unibus, and the interrupts appear as one hardware interrupt but generate 25 unique vectors starting at location 400 in the PDP-11. The error interrupt has the highest priority; the eight end-of-channel operations are next, followed by the 16 CAMAC interrupts. All 25 interrupts are on one level

and require contiguous core locations. A program plug allows one to select any Bus Request (BR) level and to start vector address.

D. CAMAC Interface

The MBD branch driver section is a very conventional design that is compatible in all respects with the CAMAC specification.^{2,3} The branch is a standard branch and can have one to seven crates. Each crate requires a standard type-A controller. The branch is terminated in the MBD and should be terminated in the last crate with a module terminator. For branches less than 10 ft long, the module terminator is not required. The maximum standard branch length is 150 ft.

The CAMAC branch has two modes of operation. The basic one is the *command mode*. The branch driver must issue a command during each branch operation. This command includes crate address information for selecting one or more crate controllers. Each addressed crate controller accepts the command from the branch highway and generates the corresponding dataway command (station number, subaddress, and function). During *read* operations, data signals are generated by a module on the dataway read lines, transferred to the branch highway data lines by the crate controller, and accepted by the branch driver. During *write* operations, the branch driver generates data signals on the branch highway, and these are transferred to the dataway write lines by the crate controller and accepted by a module. During other command operations, there is no transfer of read or write data via the branch highway.

The other mode of operation is the *Graded-L-mode* (GL) in which the branch driver addresses the on-line crate controllers and requests transfer of a composite 24-bit data word representing the state of LAM requests throughout the branch. Up to 24 different requests for attention can thus be identified in a single branch operation. Each request may represent an individual LAM condition or a combination of a number of conditions in one or more crates. In addition to this multi-level demand-handling mode, the Branch Demand (BD) line provides a single-level feature that merely indicates the presence of one or more demands, without identifying them or performing a transfer operation.

At a branch highway port, the data lines are used in the command mode for information transfers in either direction between crate controllers and the branch driver. These lines are also used to convey the pattern of requests in Graded-L mode.

Transfers in either mode through a branch highway port are controlled by interlocking timing signals that automatically adjust the timing of each branch operation

to suit the actual transmission delays and controller performance encountered. Typical branch operations require 1 to 1.5 μsec . A Graded-L operation typically requires less than 1 μsec . There is a short-cycle mode that will permit branch operation in the 600- to 800-nsec range.

The MBD branch driver has three basic registers, the 16-bit branch address register (BAR) which is decoded into the 21-bit CAMAC command register (CNAF), the 24-bit branch data register (BDH, bits 24-17 and BDL, bits 16-1), and the 24-bit graded-L-register (GLR). The F8 bit is omitted from the command word and is supplied by the processor, depending on the type of control command that is used to start the CAMAC operation. A "BCO" command is used for F8 = 0, and a "BC1" command is used for an F8 = 1 requirement.

The microprocessor has complete control of the branch driver through the control section of the branch interface. It controls reading and writing of the registers, starting branch operation, testing the "Q" and "X" lines, and testing branch busy. A branch time-out error will generate an interrupt identifiable in the CSR register. The branch "BZ" command, a 10- μsec reset command, is initiated from either a PDP-11 INIT command or a microprocessor control command. A branch demand (BD) will result in a GL operation if the processor is in the stop mode. If it is in the run mode, the processor will not allow a GL operation until it completes the current control program and executes an exit command. The result of a GL operation is that the L's from the branch are stored in the GLR and BDR registers. Bits 17-24 of the GLR result in DMA channel requests 0-7, respectively. Channel 7 has the highest priority. Bits 16-1 of the GLR are masked by the MSK register in the computer interface and appear as 16 unique interrupt vectors to the PDP-11 computer. Bit 16 is the highest priority L request. The microprocessor can test the L pattern by reading the BDR register. Bits 1-16 of the GLR can be written by the processor to test the interrupt system.

The branch is a straightforward design employing standard TTL logic. The signals on the branch are low-true.

E. Microprocessor Channel Multiplexer

1. **Basic Design.** Reviewing the MBD requirements, the modes of operation, and operation of the CAMAC and PDP-11 interfaces makes the tasks of the microprocessor clear.

The first noticeable difference or problem is the word length for the various sections. The PDP-11 is a 16-bit computer, the CAMAC command is a 17- or 21-bit word, and the CAMAC data word is a 24-bit word. The obvious choices of word lengths for the processor are 16 or

24 bits. The advantages and disadvantages of each were carefully considered before the decision was made. The most expensive single item in the MBD is the control memory. A 16-bit memory saves 50% in memory cost and is easier to load because it is compatible with the PDP-11. A 16-bit word with a specific control command will supply the 17-bit command required by CAMAC. The bus structure and arithmetic and logic unit (ALU) of a 16-bit processor are less expensive and faster.

The major advantage of a 24-bit processor is the architecture of the processor itself. With a larger word, more direct control or true microprogramming can be employed, resulting in fast operation and fewer instructions. In a 16-bit processor, more encoding-decoding is required, resulting in more logic and slower operation.

An arbitrary decision was made to design a 16-bit microprocessor. The major contributors to the decision, in order of influence, were the cost of control memory, the memory compatibility-loading problem, familiarity with the architecture of conventional processors, and the fact that at least two channel operations are required to transfer 24 bits to the PDP-11 regardless of the MBD word length.

Before word length was determined, several other problems were considered to assess their impact on the processor design. The number of words in the control memory, the number of DMA channels to multiplex, the processor bus structure, the field assignments in the instruction word, the instruction set, and the system timing were all considered separately and then jointly in arriving at the final processor design.

The major function of the processor is to transfer 16-bit words between the CAMAC interface, four registers; the computer interfaces, three registers; and the active channel registers, of which there are seven for each channel. The obvious bus structure for transferring among several registers is the two-bus structure. That is, all register outputs are gated to a source bus and all inputs are gated from a destination bus. Then all that is required is an ALU to link the two buses. The instruction that makes the transfer among registers must have fields to identify the source or destination registers. This instruction must also command the ALU to do a simple pass through; that is, transfer from a source register to a destination register. The instructions must also have fields for control functions and branching conditions. It is obvious from the field requirements that encoding is required.

Fourteen essential registers must be addressed. This requires a minimum of four bits for each source and destination field. The instruction (opc) field requires a minimum of four bits, and five would be preferable, but this leaves only three or four bits for both the control and test fields. The control field requires a minimum of four

bits to control the CAMAC and computer interfaces. The decision was made to limit the opc field to four bits and allow four bits for the control field on a certain class of instructions, and to allow the control field to be used as the test field for the branch instructions. The details of the fields will be discussed in Sec. 5.

The number of DMA channels designed into the MBD was arrived at from the requirements and an intuitive, arbitrary decision. The minimum requirements are one to two channels for experimental equipment, one for a storage display, and one for the high-speed data links. For a stand-alone system, additional channels are required for the CAMAC peripherals, such as tape units, paper-tape reader and punch, and teletypes. Because most logic and hardware operates most efficiently in multiples of four, it was decided to design for eight DMA channels and to provide expansion capability to 16 channels. As each DMA channel requires seven active 16-bit registers, cost must be considered, and this influences the decision.

At this point, the general architecture of the processor was known; but the exact instruction set, number of words in the control memory, and the system timing and speed had to be determined. The approach was to flow-chart all conceivable types of jobs and applications, then actually write the control programs with the various instruction sets and compute the system speed on the basis of an arbitrary design goal of 300-400 nsec per instruction. This goal was not completely arbitrary; to be efficient, and to overlap the operations of the CAMAC branch and the computer interface, the microprocessor had to execute three to five instructions during either interface operation. As the average time of either interface operation is 1.5 to 2 μ sec, the instruction execution time must be 300 to 400 nsec.

The decisions about the type of control memory and its size were most difficult and depended on the MBD task and mode of operation. The approach was to design for the known applications and modes while leaving the user an option as to the amount of memory his system would require. With a basic 256x1-bit memory selected, the memory increments were 256 words, with an upper limit of 1024 words determined by hardware space. The MBD addressing space will allow 4096 words of control memory. Using a 1024x1-bit memory device in place of a 256x1 unit will make this fit into the same space as the smaller memory. The initial unit will use the 256x1 memory. The memory units are read-write, random-access memory and they are TTL compatible. Read-write units were selected instead of the read-only to give the processor more power and flexibility to accomplish the wide range of requirements.

Even at the basic system design level, the interactions and trade-offs are numerous and apparent. They are even

more apparent in the more detailed discussions that follow. The one section that ties all these together, and that is the key to MBD operation, is the timing section. A crystal-controlled clock source and a four-stage, seven-state Johnson counter are the heart of the timing system. From this system, almost any arrangement of timing signals can be derived.

2. File Registers and Channel-Select Logic. The file register design is very straightforward. A set of registers is required to control a DMA-channel operation. A set is required for each channel, and the channel-select logic must connect the appropriate set to the processor. There are seven 16-bit registers per set, and there are eight DMA channels. Six of the seven in a set are general-purpose registers that can be used as the programmer desires; however, to facilitate developing an assembler for the MBD and understanding its operation, most of the registers have been assigned functional mnemonics. When a channel is selected to run, its set of file registers is connected to the source and destination buses automatically. The processor can then select the desired register by using the source and destination fields of an instruction.

To understand DMA channel operation, one should understand the functional mnemonics listed in Table III. The instruction location register (ILR) contains the address of the next command in the PDP-11 core memory list. The data address register (DAR) contains the pointer to the next word in the data list. The word-count register (WCR) contains the number of words to be transferred to

the data list. The CAMAC command register (CCR) contains the CAMAC command word minus the FB bit supplied by the type of control command executed. Two registers are general-purpose registers (GPR) that are free to be used as the programmer desires. The seventh register, the control register (CTR), has specific functions and differs from the others. A control memory entry point is required for each channel. The low 12 bits of the CTR register contain the address that will be loaded into the program counter (PC) if a jump via control register (JVC) instruction is executed. This provides a unique subroutine entry for each DMA channel. Location O of control memory will have a JVC instruction. Startup from a channel request will be through location O which will then execute a jump to the appropriate routine whose entry point is in the CTR register for that channel.

When the MBD is in the run mode and a normal instruction that has either a source or destination in control memory is executed, the address of the desired control memory word is in the low 12 bits of the CTR register. If both source and destination are selected, the same location is read and written. The upper four bits of the CTR can be tested individually using the test field of the branch instructions. This allows the programmer to select alternative procedures to execute.

Review of several control programs showed that most operations on the ILR, DAR, and WCR were either incremental or decremental and that the word was stored back in the register. To save instructions and time, a unique feature was designed into the INM (increment) and DEM (decrement) instructions if the source was a file register. If the source is a file register, it is also a destination register in addition to being the register selected by the destination field. This is true for all file registers except the CTR register. For example, the instruction increment ILR and store in MAR also stores the incremented value back in the ILR at the same time it is written into the MAR register.

The memory selected for the file registers is a 64-bit, monolithic, high-speed TTL array organized to provide 16 words of four bits. Four of the TTL memory elements are required for each type of file register. Because only eight DMA channels are being implemented, only half of the memory is being used. To prevent this waste, and because active registers are always in demand, the file register memory was divided into two banks. The bank connected to the processor is controlled by the processor. When the MBD is halted, the bank "O" is connected to ensure a known subroutine entry from the CTR register. The programmer must know with which bank he is operating. Two control commands are available for changing from one bank to the other. Both banks have the same functions. Typical access time for the memory is 33 nsec, and the readout is nondestructive.

TABLE III
FILE REGISTER OF THE MBD

<u>Mnemonics</u>	<u>Name</u>	<u>Function</u>
ILR	Instruction Location	Address of next command from core memory.
DAR	Data Address	Pointer to next word in data list.
WCR	Word Count	Number of words to be transferred.
CCR	CAMAC Command	CAMAC command word minus F8 bit.
GPR	General Purpose	Free registers.
CTR	Control	Control memory address for JVC and memory reference instructions.

The channel-select logic is the area that ties the three major areas together and is the startup logic for all run-mode operations. Understanding this area of channel requests, interrupts, exits, and priorities is the key to understanding the MBD and its capabilities.

After the processor is initialized (see Sec. III), operation is started by making a channel request. Each of the eight channels has two sources, the PDP-11 computer or the CAMAC branch. The computer request for a channel has a higher priority than the CAMAC. The eight channels have a priority of 0 to 7, with channel seven having the highest priority. When a channel is selected, the group of seven file registers associated with that channel from bank "O" are connected to the S and D buses. During the channel-priority arbitration, the program counter is reset to zero; if the source of the channel selected to run is the PDP-11, then the PC register is incremented. A minimum of 150 nsec delay between PC incrementation and the first read of the control memory allows time for the memory addressing to stabilize.

The basic processor operation is that first a channel is selected and, if its source is CAMAC, then the instruction in control memory location zero is executed. Location zero usually contains a JVC instruction that allows each channel a link to its own microprogram. If the source is the PDP-11, execution starts at location "1" of the control memory. Starting at location "1" there will be a file register initialize routine using channel transfers. This requires approximately 20 instructions for the seven file registers in a group. These will be used by each channel during system initialize to load all the file registers and will be used, as required, by channels to reinitialize during the run phase. When a transfer is completed, an end-of-block interrupt (INT) for that channel is sent to the PDP-11 by a control command. The PDP-11 recognizes the interrupt and issues a channel request, if it is desired, to reinitialize that channel.

Once the processor is executing a program, it cannot be interrupted, and it will relinquish control to the channel-select logic by executing one of four exit control commands. Four types of exits are required to establish the desired control of the four registers in the channel-select logic. The registers are the program request latch (PRL), the channel enable latch (CEL), the channel initiate latch (CIL), and the channel control latch (CCL). The PRL is used to hold a channel request while giving up temporary control to the channel select logic. The PRL can be selectively set with an EXIT 1 and selectively reset with either an EXIT 2 or EXIT 4 command. If a higher priority channel is not requesting control, control will return to the channel in the PRL. Control will eventually return even if other higher priority channels take the processor for a time. The CEL register is a mask register

for the eight-channel requests originating in CAMAC via the upper byte of the GLR register. EXIT 2 will do a selective set of CEL which is the method of enabling that channel. EXIT 3 or 4 will cause a selective rest of CEL. The CIL register is loaded from either the CSR register channel select bits, 8-10, or with a channel-initialize (CI) command. A CI command allows the MBD to change channels during a stand-alone operation. A move (MV) of the required channel number to a "O" destination and a CI control command will set that number in the CIL register and it will be in the priority structure on the next exit command. This is an indirect way of loading the CCL register. The CIL is reset by any exit if the channel that is running had a computer source. The CCL register is a three-bit register that contains the number of the currently running channel. The CCL can be read as a source but cannot be addressed as a destination. The CCL cannot be altered while the MBD is running, but only during the time between an exit and the MBD startup.

The processor operation is such that any exit causes a temporary stop. If there is no request pending, the unit remains in the stop mode until a request is made from either the computer or CAMAC. If there is a request from CAMAC (BD), a BG operation is automatically started and the channel-select logic waits for completion of the GL cycle before priority is arbitrated. The highest priority channel is loaded into the CCL register, and the MBD is given a start pulse. If the request is from the PRL or CIL, startup is immediate.

3. Control Memory. Most of the design decisions about control memory are discussed in Secs. 1 and 6. The word length was set at 16 bits to be compatible with the PDP-11, and the number of words required varies from 128 to 4096, depending on application and mode of operation. It was decided that a read/write memory was desirable for system flexibility. The memory should be TTL compatible.

The memory speed was more difficult to determine. In most systems of this type, it is desirable to match the speed of the memory to that of the central processing unit if possible. In the MBD the amount of overlap of memory and ALU operations and the desired 300-nsec instruction time are factors in determining the required memory speed. From the timing diagrams, it was decided that a 100-nsec memory was satisfactory but that a 50- to 70-nsec access time was preferable. Few memories that met these requirements and the density requirement of the selected hardware were available.

The memory that was selected has the following specifications. It is a 256-bit TTL isoplanar read/write memory with full decoding on chip. The 16-pin, dual in-line (DIP) package has a 256x1-bit organization and a 2-mW/bit

power dissipation. The device has a 50-nsec read access time and requires a 20-nsec minimum write pulse. The output is noninverting and open collector which allows easy memory expansion.

The memory increments are 256 words, and the maximum memory size is 1024 because of hardware space. The maximum memory from an addressing standpoint in the MBD is 4096 words, and it could be housed in the present hardware package by using a 1024x1 memory device. The devices are available and cost about the same as the 256-bit memory, \$0.08 per bit. The 1024 device was not used because the minimum increment would be 1024 words and many applications do not require that much control memory. Only minor MBD modifications are required to use the 1024x1 memory device. Power requirements and heating problems must be carefully assessed if the memory is expanded to 4096 words with the present system configuration.

Control memory will generally be loaded and checked in the single-cycle (SC) mode, using the load (LD) and store (ST) instructions. In SC mode, the LD and ST instructions use the PDR register as the source and destination, respectively. The memory can also be loaded or partially loaded using channel transfers in the run mode. A channel and microprogram must be dedicated to this purpose. In a stand-alone application and if the MBD has the front-panel option, the switches on the front panel are connected directly to the "S" bus, and transfer to memory from the switches is the most direct way of loading control memory. This method will be used to load a bootstrap program that will operate a paper-tape reader, data link, teletype, etc., to load the rest of the control memory. Without a front-panel option, the bootstrap can be loaded from the manual controller, but more time is required.

4. **Arithmetic and Logic Unit (ALU).** The ALU section of the MBD was very easy to design. Once the basic requirements were established, several designs from the manufacturer's literature¹⁰ satisfied the requirements.

To review requirements and decisions, the two- or three-bus processor best fits the MBD requirements. The basic ALU requirement is to transfer words from one 16-bit bus to another and to perform selected arithmetic and logic functions on the words transferred. Some of the necessary MBD functions were transfer, increment, decrement, add, subtract, logic masking, testing for zero, and shifting. The ALU should be TTL compatible. The ALU speed was fixed by the desire to have a 300-nsec instruction-cycle time. The maximum allowable time for the ALU is 50 nsec. The type of arithmetic required for the MBD is binary, positive numbers.

The ALU design, shown in Fig. 4, is very similar to the Texas Instruments design discussed in the application

notes.¹⁰ The heart of the design is a four-bit arithmetic unit with full look-ahead. The device will perform 16 binary arithmetic operations on two four-bit words. Four of the devices are required for a 16-bit ALU. The device can be implemented with active low or high data. The MBD "S" bus is low true, and the "D" bus is high true. For high-speed operation, a full-carry look-ahead scheme is made available in the ALU for fast, simultaneous carry generation. When the ALU is used with the full-carry look-ahead circuits, high-speed arithmetic operations can be performed. A 16-bit add, using full look-ahead, requires 36 nsec using TTL logic and 19 nsec using the Schottky TTL logic. Ripple carry logic requires 60 nsec for a 16-bit add. The fast look-ahead design was used with the standard TTL logic. However, the Schottky version can be used, as the devices are pin compatible.

The ALU output is stored in a shift register. The shift register (TR) output is the "B" input to the ALU unit. All arithmetic operations are on the TR register and the "S" bus. The output of the TR register, through an inverter, drives the "D" bus. By using the mode-control bits and the clock inputs to the TR register, one can implement four shift instructions.

Subtraction is accomplished by 1's complement addition in which the 1's complement of the subtrahend is generated internally. The resultant output is A-B-1 which requires an end-around or forced carry to provide A-B. The ALU can also be used as a comparator. The ALU must be in the subtract mode when making a comparison or determining relative magnitude. The A-B output is open-collector so it can be wire-AND connected to give comparison for more than four bits. The MBD is designed to test for low-byte zero and word zero using the comparison outputs. The carry output (C_{N+4}) is high if $A \geq B$ on a subtract operation. The carry output can be tested by the MBD. The most significant (DH15) and least significant (DH00) bit are also testable. The test operation (branch instructions) must follow the ALU operation that is to be tested or the information will be lost if the following instruction generates a load pulse.

5. **Microprocessor Instruction Set.** The field and word-length requirements for the instruction set were discussed in Sec. 1. The number of registers in the MBD requires a four-bit field length each for source and destination register identification. The control commands required in the branch and computer interfaces require another four-bit field. This leaves a four-bit field for the instructions codes. The ALU selection and the type of arithmetic determine the bulk of the logic and arithmetic instruction. The octal assignments and detailed programming procedures will be presented in Sec. III. Here, we discuss the not-so-obvious instructions and the design requirements that led to their implementation.

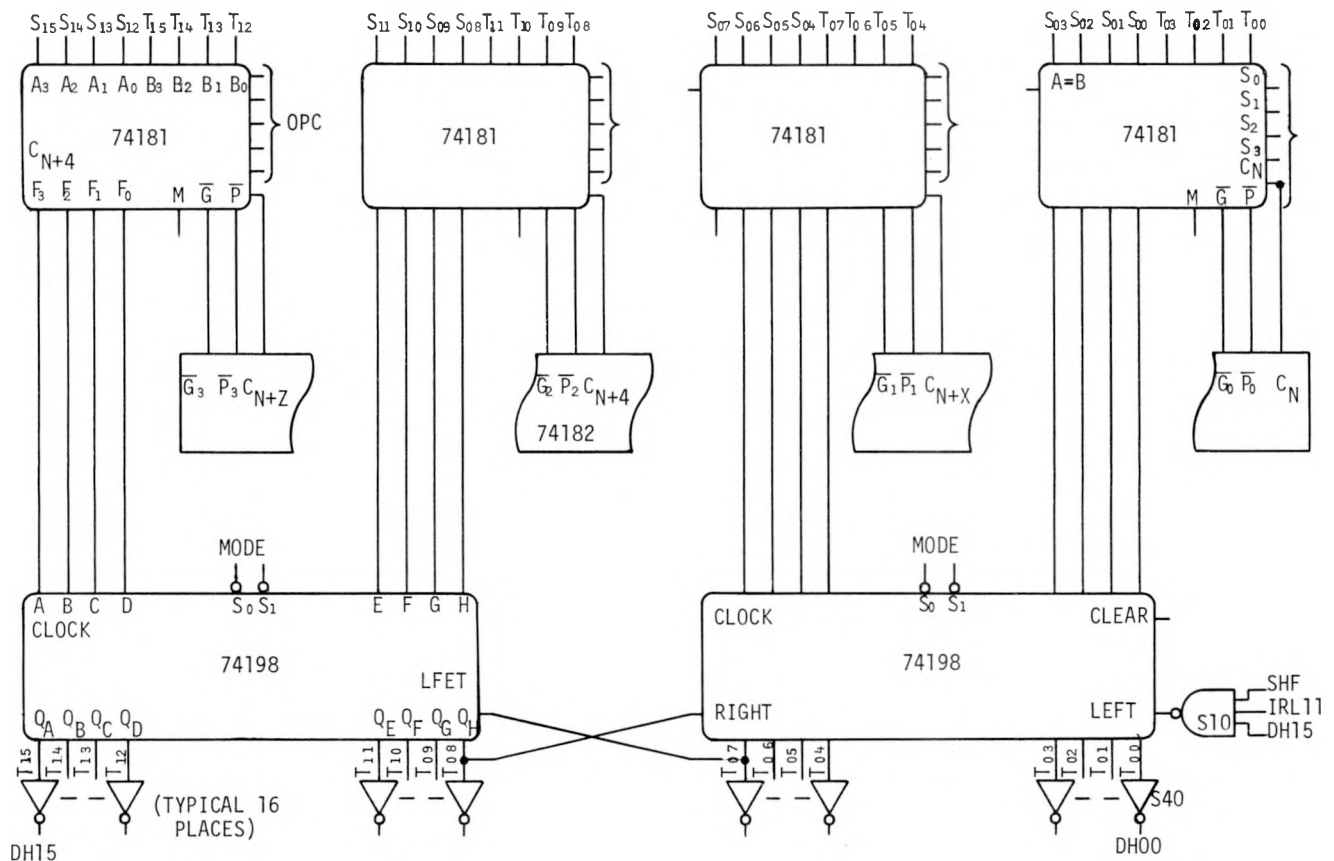


Fig. 4.
Sixteen-bit high-speed parallel accumulator with full look-ahead.

The first eight instructions are move (MV), increment (INM), decrement (DEM), add (AD), subtract (SB), inclusive OR (IOR), exclusive OR (EOR), and logic AND. These are the *normal* instructions that are implemented from the 16 available ALU instructions. A normal instruction can select one of 16 source registers and one of 16 destination registers and can specify one of 16 control commands. As explained in Sec. 2, an increment or decrement instruction with a file register source generates a destination for source registers as well as for the specified destination register. If memory is the source or destination selected, then the memory address is in the low 12 bits of the CTR register. This allows any memory register transfer and memory reference operation with any normal instruction.

The other eight instructions, the not-normal instructions, consist of the branch on condition true (BCT), branch on condition false (BCF), jump via CTR register (JVC), store in central memory (ST), load (LD) contents of memory location into TR register, load CTR from IR (LCI), jump (JP) to address specified in instruction, and the shift (SHF) instructions.

In the MBD, branch instructions are essential in controlling the two asynchronous interfaces. The branch instruction has a four-bit operations field, four-bit condition field, and eight-bit address. In a BCT, if the selected condition is true the eight-bit address is loaded into the PC counter. If the condition is false, the PC counter is incremented. With only an eight-bit address, the branches are limited to the current page of 256 words. Testable conditions are: branch demand (BD), branch busy (BRB), data channel busy (DCB), interrupt busy (INB), result negative (NF), result zero (ZF), low-byte zero (ZLB), carry bit (CF), results odd (OF), "Q" response (QF), "X" response (XF), and CTR bits 12-15. In a system dealing with asynchronous signals, the condition being tested with the branch instruction must not be allowed to change during the instruction. This is accomplished by storing all conditions with selected MBD timing signals.

The JVC instruction was invented to allow each channel a means of reaching its own microroutine. As explained in the channel multiplexer section, each CTR register contains the address (12-bit) for subroutine entry.

Once it was decided that the control memory should be read/write memory, it was necessary to have load (LD) and store (ST) instructions for reading and loading the memory. The instructions have a four-bit operation field and a 12-bit address field. If the MBD is in the run mode, the source for a store or the destination for a load is the TR register. If the MBD is in single-cycle mode, the source or destination for the LD-ST instructions is the PDR register.

The LCI instruction, load CTR from IR, was invented to save an instruction for certain control programs. It is a complement instruction to the JVC. The jump instruction (JP) is an unconditional jump. The 12-bit address of the jump instruction is loaded into the program counter. The last instructions are the shift instructions. The shifting is on the TR register, and the number of shifts is specified by bits 0-3. Bits 10 and 11 of the shift instruction specify the type of shift. If bit 10 is a zero, the direction is left; if it is a one, the direction is right. If bit 11 is a zero, the shift is logical; if it is a one, the shift is a rotate. The instructions have the following mnemonics and codes.

- SLL = 154000 Shift left logical
- SRL = 156000 Shift right logical
- SLR = 150000 Shift left rotate
- SRR = 152000 Shift right rotate

The shift instructions are the only instructions that do not have a 350-nsec cycle time. Depending on the number of shifts, the instruction varies in length from 350 nsec to over 1 μ sec. The instruction hangs the Johnson counter in state 7 until the shifts are complete, then it continues to the next cycle.

It was decided in the MBD design that a minimum amount of instruction overlap would be implemented because of the single-cycle mode of operation and simplicity of design. The price one pays for non-overlapped instruction is 50 to 100 nsec in cycle time, and the speed tradeoff for memory and ALU operation is different. For instance, the control-destination part of the cycle could be done during the next cycle. At present, the overlap is memory and ALU operation. The new memory word is being read and presented to the IR register during the ALU and control times.

6. MBD Timing System. The MBD timing diagrams and timing signals are shown in Figs. 5 and 6, respectively. The basic clock for the MBD is a 20-MHz crystal. The basic timing is derived from a four-stage, seven-state Johnson counter. The design of the Johnson counter is discussed in the Texas Instruments logic book.¹¹ Proper gating of the stage outputs provides any 50-nsec time increment.

All of the unique timing signals for the normal and not-normal instructions are shown in Fig. 5. The timing

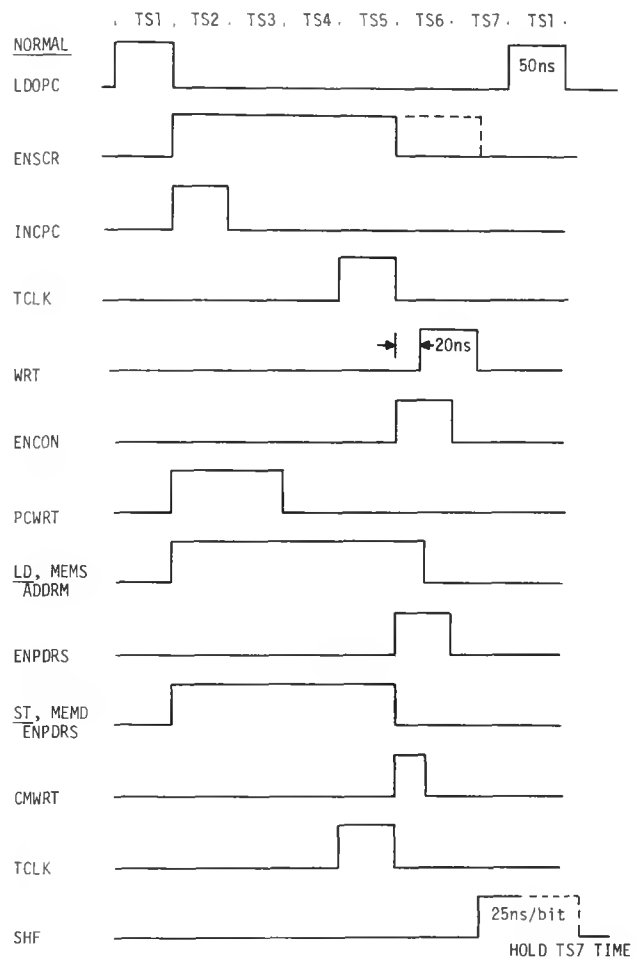


Fig. 5. MBD timing diagram.

for a normal instruction is at TS1 where the instruction is loaded into the IR register and decoded. At TS2, the source is enabled to the S bus and the PC counter is incremented. During TS2 to TS4, the ALU operation takes place, and at TS5 the ALU output is loaded into the T register with the TCLK and gated to the D bus. At TS6, the data on the D bus are loaded into the selected destination register. If a control command is specified, TS6 is the 50-nsec control pulse. TS7 is a dead cycle or the halt state in single-cycle mode. In the run mode, this time is required for memory access.

The not-normal instructions that have unique timing requirements will be discussed separately. The BCT, BCF, JVC, and JP instructions require only an increment of the program counter (INCPC) or a parallel load of the PC counter (PCWRT). In a parallel load of the PC counter, the source is either the IR register or the CTR register. A shorter timing cycle could have been implemented for

F. MBD Manual-Control Console

The control console, a separate unit, was designed to facilitate checkout and testing and to control initializing of the MBD as a stand-alone branch driver. Figure 7 shows the front panel of the MBD manual-control console.

Console operation is that of a PDP-11 simulator, and it connects to the MBD through the Unibus connector. The console should not be connected to the MBD at the same time as the PDP-11. The unit has 16 switches for entering data or addresses into their respective registers. The registers can be loaded either from the console switches or from the Unibus, depending on the type of transfer executed. Lights on the console display the contents of the data and address registers.

The manual controller will be used primarily for program I/O transfers (DATO and DATI). The sequence for a DATO transfer is as follows.

1. Load data to be transferred into the console switches and load in the data register by pressing the load data switch.
2. Load address of device in address register by pressing load address switch.
3. Set the read/write switch to write position, then start the DATO operation by pressing the program transfer switch.

The DATI operation is as follows.

1. Load the address of device in address register.
2. Set read/write to read and press transfer switch. The data transferred from the device will appear in the register and data lights.

The console has an INIT switch that is the master clear for the system.

The operation of both types of priority transfers can be tested. For a bus request (BR), a light on the console indicates the request and pressing the bus-grant (BG)

switch allows the transfer to be completed. The interrupt vector will appear in the data lamps.

The nonprocessor request (NPR) is basically the same as the BR. The NPR lamp indicates the request, and pressing the nonprocessor grant (NPG) switch completes the cycle. One difference is that if the data are transferred from the console, they must be loaded into the data register before the grant is initiated. If the transfer is to the console, the data will appear in the lamps. Generally, transfers and interrupts will be initiated in the CAMAC hardware. A channel request can be initiated with a program transfer to the CSR register in the MBD from the control console.

G. MBD Logic Selection

The MBD logic requirements were heavily influenced by the speed and compatibility requirements of the CAMAC branch and the PDP-11 computer. Other considerations were type of hardware package and density of package from a noise and heat standpoint. Speed-power tradeoffs, cost, and designer confidence and familiarity with logic families contributed to the final decision.

The selection of the transistor-transistor logic (TTL) family was an obvious choice. However, the choice of series within the family was not so obvious. The four members of the family are the standard series, the high-speed series, the low-power series, and the Schottky diode-clamped series. All four series are compatible and will interface directly with each other. Typical characteristics of the family are:

Supply voltage	5.0 V
Logical 0 output voltage	0.2 V
Logical 1 output voltage	3.0 V
Noise immunity	1.0 V

The standard series offers a combination of speed and power dissipation best suited for most applications. The

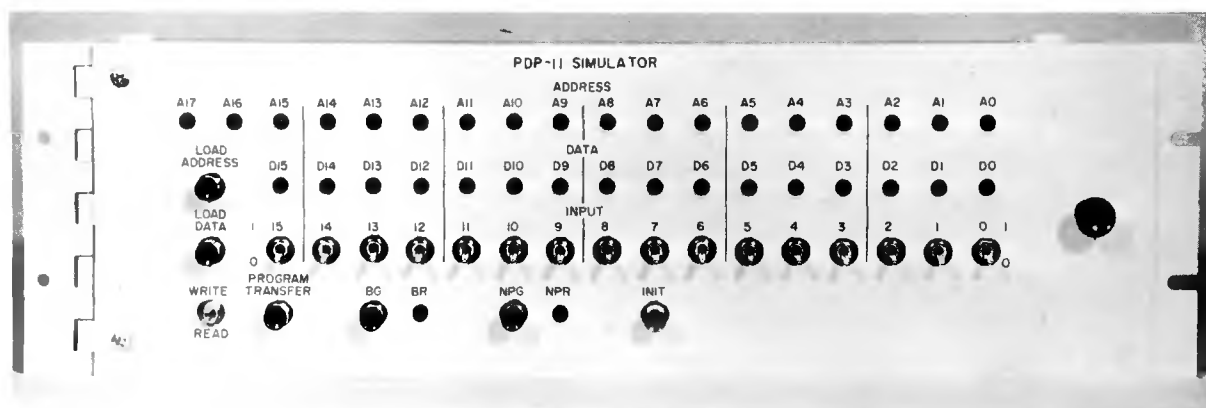


Fig. 7.
MBD manual-control console.

multiple-emitter input transistor offers the most logic for the least physical size, and it is a major contributor to the fast switching speed of TTL. Low output impedance is attained with the active pull-up output transistor, which also provides improved noise immunity and faster switching. A speed of 10 nsec per gate is typical.

The low-power series has one-tenth the power dissipation of standard logic. Power dissipation of 1 mW per gate and a speed of 33 nsec per gate are typical.

The high-speed series is basically the same as the standard series. Smaller resistors are used, and the output consists of a Darlington transistor pair. Because of the low impedance output and the transistor action, a higher speed of 6 nsec per gate can be obtained. However, as a disadvantage, more power is required.

The Schottky-clamped TTL series has the highest speed in the family, equivalent to the unsaturated logic (ECL) with the relatively low power consumption of TTL. Typical gate delays are 3 nsec, and average power dissipation per gate is 20 mW.

The decision was to use standard TTL series for the CAMAC and computer interfaces and to use the Schottky series where higher speed is required, primarily in the microprocessor section. A typical standard series IC costs \$0.20 to \$0.50, and a typical high-speed series costs \$1.00 to \$1.50. When the cost of the low-power series is lower and the variety of function is higher, the series will be more competitive with the standard series.

Many of the more complex functions (MSI) have already been discussed. In particular, the memory and ALU integrated circuit selected had a large impact on the rest of the MBD design.

H. MBD Hardware

Both the MBD and the manual-control console were designed and built using the computer automated system hardware (CASH) developed by Standard Logic, Inc. The CASH hardware is designed for use with computer-aided design technology and is supported by a complete line of systems software, computerized documentation, and semiautomatic wiring. The eight-channel, 1024-words-of-memory model of the MBD, with front-panel option, has 465 integrated circuits and requires a 5-V dc, 25-A power supply.

The MBD's size and complexity, speed of operation, power requirements, heating, and processor architecture were all considered in the hardware selection. The quantity of units to be built, the Laboratory's in-house capability, the type and availability of technical assistance, the number of design options, and the design uncertainties require hardware that is easily modified and can be mass produced with minimum effort. The choices

considered were wrap systems (CASH), logic cards, large printed-circuit card design, and building within the PDP-11 computer hardware.

The wire-wrap structure has the highest density, resulting in lower capacitance and shorter buses in the microprocessor. It can be fairly easily modified, and, with the computer-aided wiring, can be mass produced, with many options, using a minimum work force. Also, the documentation is immediate and correct for the many changes and options. The hardware requires IC sockets that are expensive and can cause contact problems. Because of the high density, heating can be a problem and there is the added cost of cooling.

With standard logic cards, the system would be very large and expensive. It is doubtful if the system would have worked at the design speed because of the wire length and large capacitance. This approach was not seriously considered.

The larger printed-circuit (PC) card approach has some strong points: easy mass production, reliability, repeatability (less checkout), and low cost for larger quantities. This approach has been a major factor in low-cost minicomputers. The disadvantages are: layout costs, inexactness of design, necessary prototypes, loss of efficiency from many option board layouts, and the fact that design changes of any magnitude require another layout. It was decided that the quantity of MBD's required and the time required to develop a PC system were not justified.

Using the computer hardware has some attractive features. The structure of the PDP-11/20 computer is fairly large PC cards, and the package, system units, will accept multiple standard logic cards. In theory, with this approach the MBD could have been designed on a few PC cards and inserted into the system units of the PDP-11 as simply another computer interface. Much of the logic for the computer interface section of the MBD was available from the computer manufacturer. However, this approach was not taken because space may not always be available in the computer, the family of PDP-11's do not all use the same type of hardware, the MBD power requirements are very large and would usually overload the computer supply, the system could not be used as a stand-alone processor, and all of the disadvantages of the larger PC card approach apply.

It was decided to use the wire-wrap system (CASH) because of the high-density, computer supported, easy modification; the many options; and the fact that a stand-alone device could be designed. The CASH system has a variety of cards using 14-, 16-, and 24-pin sockets for IC's, and the wiring is a two-level wire wrap using No. 30 wire. A third level can be added by hand wiring if necessary. The MBD has 4300 wires, and was wired in less than a week using the machine-aided wiring system.

The MBD was designed in a drawer, which has a 3½-in. panel height, slides, and two cooling fans. The drawer has one 16- by 15-in. wire-wrap plane that has 18 CASH card positions for the microprocessor logic. The front panel carries the CAMAC port connector, and, if the front-panel option is required, the program counter and destination bus are displayed on lights and 16 switches are added to input directly to the source bus. The rear panel carries the Unibus connector and connector for the separate power supply.

The manual controller uses the CASH hardware and is housed in a 5-in. vertical panel. The unit has three CASH cards and a 5-V dc, 5-A power supply. A standard Unibus cable connects the unit to the MBD. The control panel contains the control switches and the address and data lights.

III. OPERATION OF MBD

A. General Operation

The design section discussed the MBD hardware. This section is somewhat redundant, but presents the MBD from a systems programmer's point of view. The intent is to present the procedures and details of MBD operation that will allow a programmer familiar with the PDP-11, the CAMAC branch, and basic MBD operation to generate the microprograms required for the MBD. In a true microprogrammed device, the programmer must be familiar with every circuit and gate. This is not required in the MBD; but a thorough understanding of the operating modes, the internal registers, and the instructions set is necessary.

The following general systems procedure will point out the considerations and decisions necessary for writing the MBD codes. The system configuration should be documented and analyzed to determine the MBD's mode of operation. The data rates, event rates, number of channels, amount of preprocessing, channel priority structure, and interrupt levels determine how much time the MBD has and what modes must be implemented to satisfy the system requirements.

From the computer end, the systems engineer has to consider whether he is going to operate the MBD-controlled branch under the disk operating system (DOS). The amount of core, the location of the magnetic tape and storage display interfaces, the disk size and speed (assuming that the system has a disk), and the other peripheral requirements all enter into the requirement for the number of DMA channels in the MBD, the priorities assigned to the channels, and the MBD's mode of operation. For example, if the magnetic tape interface is in the

CAMAC branch, then it may not be necessary to transfer data to the PDP-11 for data-logging operation. If the CAMAC command lists are short and the MBD has enough memory, higher speed can be obtained if the lists are in the MBD memory rather than the PDP-11 memory. The number of channels and the interrupt requirement determine the PDP-11 system and programming requirements. Typical codes required are interrupt handler, data analysis, store program for raw and analyzed data, plot routines, load MBD memory, initialize MBD channels in run mode, and other peripheral controllers.

From the CAMAC branch, it is essential to know the number of crates, crate numbers, and distances from the MBD. The last crate should have a terminator. The type and location of every module and the commands for each are required. The general operation of each module, the LAM structure, how the module responds to the inhibit, and the speed of operation are all inputs into assigning the module to a particular channel of operation. This will indirectly determine the LAM patching on the Type-A controllers if the systems do not have LAM grader modules.

After reviewing the experimental requirements of the PDP-11 and the CAMAC branch, one can determine the number of channels, what operations on what priorities, and the mode of MBD operation. At this point, one can start to think of coding the MBD. The first step will generally be to make a flow diagram for the various operations and try to estimate the necessary amount of memory and speed of operation. The details of the registers and the instruction set will be presented in the following sections. Examples of PDP-11 programs for loading the MBD memory and initializing the MBD and CAMAC systems are presented. Examples of MBD microprograms for channel initializing, fast data acquisition, and bootstrapping for loading MBD from paper tape are also presented to show the technique and format of the symbol-definition macroassembler. The assembler defines the fields of Table IV and will sum the octal numbers from the mnemonics to form the MBD microinstruction. The absolute, sequential list of instructions can then be transferred from the PDP-11 to the MBD or loaded using the bootstrap operation of the stand-alone system.

Extensive diagnostic programs have been written for developing and testing the MBD. In addition to the examples shown here, programs have been written for every known mode of MBD operation. Therefore, coding the MBD becomes a problem of familiarizing oneself with existing codes and system operation, making the system decision, and selecting the appropriate set of programs. The programmer will have to make the CAMAC command lists and modify the existing codes, but this is a simple operation.

TABLE IV
OCTAL REPRESENTATION OF INSTRUCTIONS

<u>Operation</u>	<u>Control or Condition</u>	<u>Source</u>	<u>Destination</u>
000000 MV	(None)	0000 BD	000 SWS
010000 INM	RDR	0400 DCB	020 ILR
020000 DEM	WTR	1000 BRB	040 DAR
030000 AD	RDH	1400 INB	060 WCR
040000 SB	WTH	2000 NF	100 CCR
050000 IOR	BCO	2400 ZF	120 CTR
060000 EOR	BC1	3000 ZLB	140 GP1
070000 AND	EX4	3400 CF	160 GP2
100000 BCT	EX1	4000 QF	200 PC
110000 BCF	EX2	4400 XF	220 MDR
120000 JVC	EX3	5000 OF	240 MAR
130000 ST	INT	5400	260 BDL
140000 LD	CI	6000 C12	300 BDH
160000 LCI	BKO	6400 C13	320 CCL
170000 JP	BK1	7000 C14	340 PDR
154000 SLL	BZ	7400 C15	360 MEM
156000 SRL			17 MEM
150000 SLR			
152000 SRR			

B. Description of Registers

1. CPU Registers. Four registers addressable by the computer central processor unit (CPU) control communication between the computer CPU and the MBD processor. All four are 16-bit registers with device addresses (DA) from 764000₈ to 764006₈.

CSR: Control and Status Register; DA 764000₈, write only (bits 0-6 and 8-10) or read only (bits 7 and 13-14). Used to issue control functions such as "Reset MBD" or "Initialize Channel" and to indicate the status of various conditions. When the CSR is loaded with single-cycle mode specified (or bit 1 = Reset MBD), the MBD ceases all previous operations and executes the CSR function in one cycle and halts. When the CSR is loaded with bit 2 and the run mode is specified, the MBD continues its current program (if any) to the next EXIT command, the Program Counter (PC) of the MBD is then set to location 1, and the program stored in the control memory (CM) of the MBD is executed starting from location 1, with the active channel selected by bits 8-10 of the CSR. Note that channel interrupts from CAMAC will bypass this set of instructions which start at location 1 because such interrupts branch to location 0, which should contain a Jump Via CTR (JVC) instruction to branch to the beginning of the appropriate service routine.

The bits in the CSR have the following functions.

Bit 0: Mode of operation. Normal operation with a computer is in run mode; single-cycle mode is reserved for initialization and manual operation (0 = run mode, 1 = single-cycle mode).

Bit 1: Reset MBD. Clears all channel-request latches (CIL, CEL, PRL, and CCL), sets the program counter (PC) to 0, and sets the mode to single-cycle regardless of bit 0 of the CSR. The Ready bit (bit 7) of the CSR is set at the end of the MBD cycle. A Reset MBD instruction should precede the first Initialize Channel instruction to reset the PC to 0.

Bit 2: Initialize Channel. Bits 8-10 of the CSR set the Channel Initialize Latch (CIL) to the desired channel. The Ready bit (bit 7) of the CSR is set, and the CIL is reset when the channel is accepted by the MBD, i.e., when the initialization routine for that channel begins at location 1. This instruction increments the PC to point to location 1, and it therefore assumes that the PC has been previously reset to 0 by a Reset MBD instruction from the computer CPU or by an EXIT instruction in the MBD program.

Bits 4-5: Extended Memory. Not yet implemented.

Bit 6: Interrupt Enable. Enables the program interrupt (at priority BR5 with vectors starting at location 400₈ in computer memory) to the computer CPU for the 25 interrupts of the MBD (0 = disable, 1 = enable). The highest priority (at the highest location in computer memory 540₈) is the branch or bus error, which is identifiable with bits 13 and 14 of the CSR. The next eight interrupts are from INT commands from the eight MBD channels, and the lowest priority interrupts are the low-order 16 bits of the graded-LAM requests (GLR) masked by the MSK register.

Bit 7: Ready. Status of MBD (bit 1) or initialized channel (bit 2). Note that bit 7 is easily testable by byte tests for sign.

Bits 8-10: Channel Select. Selects which of eight channels to initialize by bit 2.

Bit 13: Branch Error. A branch time-out error will generate an interrupt identifiable by this bit.

Bit 14: Bus Error. A Unibus transfer time-out error will generate an interrupt identifiable by this bit.

PDX: Program Data Register; DA 764002₈, read-write. Used to transfer data or instructions between the computer CPU and the MBD registers or control memory (CM).

MSK: Mask for GLR interrupts; DA 764004₈, read-write. Used to mask (AND) the low-order bits (1-16) of

the graded-LAM (GL) interrupt requests (GLR). A bit = 1 will enable an interrupt at the corresponding priority. An INIT instruction clears the mask register.

IR: Instruction Register; DA 764006₈, write only. Used to load the Instruction Register (IR) of the MBD from the computer CPU and execute the instruction. The MBD should be in single-cycle mode for proper operation.

2. DMA Registers. Two 16-bit registers are addressable only by the MBD but control the flow of information on the Unibus to the CPU.

MAR: Memory Address Register. Contains the address in computer memory for direct memory access (DMA) transfers. This address is a *word* address, whereas addresses transferred from PDP-11 programs are generally *byte* addresses and must be divided by two (or shifted right one bit).

MDR: Memory Data Register. Contains the data of the last DMA transfer from memory or the next transfer to computer memory.

3. CAMAC Registers. Three registers are addressable by the MBD and control communication between the MBD and the CAMAC crates and modules.

BAR: Branch Address Register. Sixteen bits decoded into twenty-one bits. Used to specify the Crate, Number, Address, and Function (CNAF) commands for CAMAC operation.

Bits 0-3: Address (A) or register within module. Transferred as four bits BA1, 2, 4, and 8 to CAMAC.

Bits 4-8: Number (N) of module within crate. Transferred as five bits BN1, 2, 4, 8, and 16 to CAMAC.

Bits 9-11: Crate number (C) between 1 and 7. Decoded to one bit of seven bits BCR1-BCR7 for CAMAC with one bit per crate.

Bits 12-15: Function (F) to be performed, not including F8, which is specified by the control (CTRL) portion of the instruction register (IR) by BCO (F8 = 0) or BC1 (F8 = 1). Decoded into five bits BF1, 2, 4, 8, and 16, including F8, to CAMAC.

BDL and BDH: Branch Data Register. Twenty-four bit register divided into two parts for manipulation by sixteen bit processor. BDL: low-order bits (1-16) of CAMAC data. BDH: high-order bits (17-24) of CAMAC data.

GLR: Graded L's Register. Twenty-four bit register of which only the low-order portion is addressable by the MBD. GLR: low-order bits (1-16) of the Graded LAM (GL) requests. Computer CPU program interrupts from these bits are masked with MSK. GL Channel Requests: high-order bits (17-24) of the GL requests are interpreted directly as requests to the corresponding eight channels (0-7) of the MBD, with channel 7 highest priority and channel 0 lowest priority.

4. File Registers. Each of the MBD's eight channels has two banks of seven registers (all 16-bit registers) to control the operation of each channel (14 registers per channel). The Channel Control Latch (CCL) determines which set of file registers (which channel) is active. The banks are selected by the appropriate control (CTRL) function in a normal instruction. Normal operation is through the registers of bank 0, which is selected by any EXIT operation (so that each interrupt begins in bank 0). The registers of bank 1 are available for auxiliary operation. Five of the registers are nominally assigned specific purposes, but only the CTR is restricted to specific functions. All may be specified as the source (SRC) or destination (DST) of a normal instruction.

CTR: Control Register. Contains status bits and the program address in control memory (CM) for interrupts and for addressing the CM by normal instructions. Bits 0-11 of the CTR may be loaded directly from the instruction register (IR) by an LCI instruction, as well as being specified as the SRC or DST of a normal instruction.

Bits 0-11: Program address in CM used by normal instructions with CM as SRC or DST or by Jump Via CTR (JVC) instruction to begin processing an interrupt at the specified address. High-order bits are ignored for CM less than 4096 words.

Bits 12-15: Status bits that may be set and tested by MBD instructions.

ILR: Instruction Location Register. List pointer that contains the address of the next word in the list of instructions stored in the computer memory. Used to transfer instructions from the computer memory to the MBD processor.

DAR: Data Address Register. List pointer that contains the address of the data list stored in computer memory. Used to transfer data between the MBD and computer memory.

WCR: Word Count Register. Contains the running count of the number of words to be transferred in a block transfer of data.

CCR: CAMAC Command Register. CAMAC CNAF command as for the BAR with 16 bits, not including F8.

GP1: General-Purpose Register Number 1. For general use.

GP2: General-Purpose Register Number 2. For general use. Note that the seven auxiliary registers in bank 1 can be used as additional general-purpose registers (though addressed as specific file registers). The CTR in bank 1 functions like the CTR in bank 0.

5. MBD Registers. A number of special-purpose registers are addressable in whole or in part by the MBD processor.

CM: Control Memory. Nominally 1024 words (minimum 256, maximum 4096) of 16-bit memory-containing program instructions and data for MBD operations. Addressable either by the store (ST) and load (LD) instructions or as the source (SRC) or destination (DST) of a normal instruction.

IR: Instruction Register. Sixteen-bit register that contains the instruction to be executed by the MBD. The IR may be loaded from the computer CPU by loading the IR, after which the instruction is executed in single-cycle mode. In normal run-mode operation, the IR is loaded automatically in sequence from the address in the CM specified by the program counter (PC).

TR: Arithmetic Register. Sixteen-bit register that contains the result of the last normal operation. Its value is included as a source in many of the normal instruction operations. The shift (SHF) instruction operates directly on the TR without using the arithmetic logic unit of the MBD processor, so the tests for zero and carry are not valid, though the test for negative is. For all other instructions, the TR and the arithmetic logic unit can be considered synonymous.

PC: Program Counter. Twelve-bit register pointing to the address in the CM of the next instruction to be executed. The PC is set to 0 by the Reset MBD function (bit 1) of the CSR or by any EXIT command, loaded from the channel CTR by a JVC instruction, incremented by the Initialize Channel function (bit 2) of the CSR or by execution of any IR instruction in single-cycle mode, incremented automatically during normal instruction

sequencing, set by the JP instruction, or set or incremented by branch instruction BCT or BCF. It may also be read as a source (SRC) of a normal instruction, but not altered by writing as a destination (DST). Note that the PC of the MBD points to *word* addresses, whereas the PC of the PDP-11 computer CPU points to *byte* addresses.

CIL: Channel Initialize Latch. Eight-bit register (of which only one bit may be set at a time) used to initialize any one of the eight channels. It is loaded from bits 8-10 of the CSR (with bit 2 set) or by the Channel Initialize (CI) Command and is reset by any EXIT command in that channel, or by a RESET MBD function (bit 1) of the CSR.

CEL: Channel Enable Latch. Eight-bit register (one bit per channel) specifying which channels have hardware interrupts enabled. An EXIT 2 command will selectively set (enable) the appropriate bit of the CEL corresponding to the channel of the top eight bits of the GL (if CAMAC) or the channel being initialized (if computer CPU). An EXIT 3 or EXIT 4 command selectively resets (disables) the corresponding bit.

PRL: Program Request Latch. Eight-bit register (one bit per channel) specifying which channels have programs waiting to be executed. An EXIT 1 command will set the appropriate bit in the PRL corresponding to the channel initiating the interrupt being serviced (if CAMAC) or the channel being initialized (if computer CPU). An EXIT 2 or EXIT 4 command selectively resets the corresponding bit. Setting a bit in the PRL is equivalent to issuing a hardware interrupt request to the corresponding channel with the channel enabled. When no higher priority channels are active or requesting service, that channel is selected and the instruction (JVC) in location 0 of the CM is executed to branch to the service routine for that channel.

CCL: Channel Control Latch. Eight-bit register (of which only one bit may be set at a time) specifying which channel is currently in control and active. The CCL may be read as a SRC, but not altered by software as a DST. The CCL is set to a specific channel whenever that channel is being initialized by the computer CPU (by loading the CSR with bit 2 and the channel number in bits 8-10), or a hardware interrupt to that channel (from CAMAC GL bits 17-24) is accepted and service is begun. The CIL can be loaded with a CI command, and the channel number loaded into the CIL will be in the priority request at the next EXIT command. This is an indirect CCL load which is required for stand-alone systems in order to change channels.

C. Instructions

1. Normal Instructions. Normal instructions consist of four parts. The operation (OPC) is specified by bits 12-15; the source (SRC) and destination (DST) registers are specified by bits 4-7 and 0-3, respectively; and an optional control (CTRL) command is specified by bits 8-11. The SRC and/or DST can be omitted (NON = 00) for operation with the TR; the result of any operation (by a normal instruction) is always left in the TR as well as the DST. The octal representations of all instructions and elements are given in Table IV; a complete instruction is the sum of the appropriate codes, one from each column. Some mnemonics have been changed from their original definitions for consistency and compatibility with PDP-11 software; SRC and DST mnemonics may be distinguished by adding prefixes of S and D, respectively. If the SRC or DST is control memory, then the address is in bits 0-11 of the CTR.

MV: Move SRC to DST and execute CTRL.

INM: Increment SRC, store the result in both SRC (only if it is a file register and not the CTR) and DST, and execute CTRL. The CARRY flag is set on overflow from 177777_8 to 0.

DEM: Decrement SRC, store the result in both SRC (only if it is a file register and not the CTR) and DST, and execute CTRL. The CARRY flag is set on AD underflow from 0 to 177777_8 .

AD: ADD 2's complement TR to SRC, store the result in DST, and execute CTRL. The CARRY flag is set on overflow if $SRC + TR > 177777_8$.

SB: Subtract TR from SRC, store the result in DST, and execute CTRL. The CARRY flag is set if $SRC \geq TR$.

IOR: Inclusive OR, TR, and SRC. Store the result in DST, and execute CTRL.

EOR: Exclusive OR, TR, and SRC. Store the result in DST, and execute CTRL.

AND: AND, TR, and SRC. Store the result in DST, and execute CTRL.

2. Branch-Conditional Instructions. Branch-on-condition instructions have three parts. The OPC is specified by bits 12-15, the condition (COND) to be tested is specified by bits 8-11, and the address (ADDR) in the CM for the branch is specified by bits 0-7. Because the

maximum address specified is 8 bits, branching may be done only within the current page of 256 words. The TR and conditional flags are not affected by branch instructions.

BCT: Branch to ADDR if COND is True-Continue, with $PC = PC + 1$ if COND false.

BCF: Branch to ADDR if COND is False-Continue with $PC = PC + 1$ if COND true.

3. CM Addressing Instructions. The Control Memory (CM) addressing instructions have two parts: The OPC is specified by bits 12-15, and the address (ADDR) in the CM is specified by bits 0-11 (except for the JVC and LCI instructions).

ST: Store TR into CM at ADDR. If mode is single-cycle, the source is the PDR rather than the TR.

LD: Load the contents of ADR in CM into TR. If mode is single-cycle, the destination is the PDR.

JP: Jump via ADDR. Deposit ADDR into the PC, effecting a jump to the address specified by the IR. The TR is not affected.

JVC: Jump via CTR. Transfer bits 0-11 of the appropriate CTR (specified by the CCL either by the interrupting channel or the channel being initialized) into the PC, effecting a jump to the address specified in the CTR of the appropriate channel. Note that JVC should be the first instruction of the MBD program (in location 0) so that hardware interrupts, which branch to location 0, can jump to the service routine appropriate for the interrupting channel.

LCI: Load CTR from IR. Transfer bits 0-11 of the IR to the CTR, providing a pointer for interrupts to the address specified by the IR. Note that the LCI instruction does not directly address the CM or affect the PC, but prepares the CTR for eventual use by the PC, or by a normal instruction with the CM as an SCR or DST. It does affect the TR, loading it with bits 0-11 of the IR.

4. Shift Instruction. The shift instruction has three parts. The OPC is specified by bits 12-15, the number of bits to be shifted is specified by bits 0-3, and the direction and type of shift are specified by bits 10 and 11.

SHF: Shift TR N bits right/left and rotate/logical.

Bits 0-3: N (0-15). Number of bits TR is to be shifted.

Bit 10: Direction. 0 = left, 1 = right.

Bit 11: Type. 0 = rotate, 1 = logical.

SLL = 154000	Shift left logical
SRL = 156000	Shift right logical
SLR = 150000	Shift left rotate
SRR = 152000	Shift right rotate

5. Control Commands. The control command (CTRL) is specified by bits 8-11 of a normal instruction. It is normally executed as part of (and after in timing) a normal instruction. Note that an instruction consisting of a CTRL command only is essentially a MV NONE, NONE, and CTRL instruction and its only effect besides the CTRL execution is to clear the TR. The TR is unaffected if the SRC and DST are left blank for AD, IOR, EOR, or AND instructions.

RDR: Computer Memory Read and Release. Read one word from computer memory at MAR into MDR and release the Unibus after transfer is complete.

WTR: Computer Memory Write and Release. Write one word from MDR into computer memory at MAR and release the Unibus after transfer is complete.

RDH: Computer Memory Read and Hold. Read one word from computer memory at MAR into MDR and hold the Unibus (retain control as master) after transfer is complete. The Unibus can be released by a RDR or WTR instruction or by any EXIT command.

WTH: Computer Memory Write and Hold. Write one word from MDR into computer memory at MAR and hold the Unibus (retain control as master) after transfer is complete. The Unibus can be released by a RDR or WTR instruction or by any EXIT command.

BCO: Branch Control Command. Issue to CAMAC the CNAF command stored in the BAR with F8 = 0 (normal read/write commands).

BC1: Branch Control Command. Issue to CAMAC the CNAF command stored in the BAR with F8 = 1 (control commands).

BZ: Branch Zero Command. Issue to CAMAC the BZ command to clear and initialize. The PDP-11 hardware reset clears the CAMAC branch with a BZ command, but does not reset the MBD.

EX1: EXIT at priority 1. Suspend program execution for the channel in progress and set the corresponding bit

of the PRL. When no higher priority channels are active or requesting service, that channel is selected again and the PC is set (by a JVC instruction at location 0) to the location in CM specified by bits 0-11 of the CTR of that channel. Used to relinquish control temporarily to higher priority channels if they are requesting service.

The EXIT commands provide an opportunity for interrupt requests to be serviced. While a program is running in the MBD, it cannot be interrupted, except by a Reset MBD command from the computer CPU. Any EXIT command terminates program execution, resets the PC to 0, selects bank 0, examines the priority of all interrupt requests, and begins to service the interrupt request from the channel with the highest priority.

If a program running at low priority wants to allow higher-priority requests to interrupt, it may issue an EX1 command (after setting the CTR to point to the next instruction), which sets the PRL bit representing a software request to interrupt at that channel. Program execution will be suspended temporarily. As soon as all (if any) higher-priority channel requests (hardware or software) have been satisfied, program execution will continue (at the location specified by the CTR).

When a program is finished, but more hardware requests on that channel are expected, an EX2 command will set the CEL bit to enable CAMAC Graded-LAM requests (GL) on that channel to request an interrupt. The PRL bit is reset so that no request appears on that channel until the hardware request from the GL. An EX4 command prevents any request on that channel from appearing until the channel is re-initialized. An EX3 command disables hardware (CEL) requests, but leaves the software (PRL) requests unchanged from their previous settings by EX1 or EX2 commands.

EX2: Exit at priority 2. Suspend program execution for the channel in progress, reset the corresponding bit of the PRL, and set the corresponding bit of the CEL. Program execution for that channel will commence at the address specified in bits 0-11 of the CTR for that channel when a hardware interrupt on the corresponding bit from the CAMAC GL (bits 17-24) is requested and no higher priority channels are active or requesting service. Used at the end of an interrupt servicing routine.

EX3: Exit at priority 3. Suspend program execution for the channel in progress and reset the corresponding bit of the CEL. Interrupt requests from the CAMAC GL (bits 17-24) will be ignored. If the corresponding bit of the PRL is already set (by a previous EX1), EX3 will act like EX1; otherwise, it will act like EX4. Used to disable interrupts to a channel.

EX4: Exit at priority 4. Suspend program execution for the channel in progress and reset the corresponding bits of both the PRL and the CEL. Program execution for that channel is terminated until that channel is re-initialized by the computer CPU.

INT: Interrupt. Issue a program interrupt to the computer CPU at the interrupt vector associated with the channel in progress (as determined by the CCL). The eight channels (0-7) correspond to high-priority interrupts (17-24); only one channel can have an interrupt executed at one time.

CI: Channel Initialize Latch SET. Set the appropriate bit of the CIL from the TR. Pertains to stand-alone operation only. The instruction MV SRC, CI, loads the CIL with the number in bits 0-2 of the SRC.

BKO: Specify Bank 0 file registers. Normal operation of the special-purpose file registers (CTR, ILR, DAR, WCR, and CCR) is with the registers of bank 0. This set may be specified by either the BANK 0 or any EXIT control command. Hardware interrupts may assume bank 0 because the last command before an interrupt is accepted for service can only be an EXIT command.

BK1: Specify Bank 1 file registers. Each channel has an auxiliary set of seven file registers with the same SRC and DST names as those of bank 0, but available for any purpose. Note that instructions operate equally well on the registers of bank 1 and bank 0, depending on which bank is operative.

6. Condition Tests. The condition test (COND) is specified by bits 8-11 of a branch-conditional instruction (BCT or BCF). Note that the tests for zero and carry (ZERO, ZERO LO, and CARRY) are not operative (always reset) after a SHF instruction, which operates directly on the TR without using the arithmetic logic unit. A null instruction after the shift (e.g., AD, IOR, or EOR with no SRC or DST) will set the zero flags properly for testing, but a test for carry after a SHF instruction is not possible.

DCB: Data Channel Busy. True if the Unibus data channel is busy.

BRB: Branch Busy. True if the CAMAC branch line is busy.

INT: Interrupt Busy. True if a computer CPU interrupt is already present and not yet satisfied.

BD: CAMAC Branch Demand. True if a CAMAC interrupt request is present.

ZF: TR Zero word. True if the entire word of the TR is zero.

ZLB: TR Zero Low Byte. True if the low byte of the TR is zero.

CF: Arithmetic CARRY. True if the last arithmetic operation resulted in a carry (overflow). See the description of normal instructions for details.

QF: Q Flag of CAMAC. True if a Q response was generated by the last CAMAC command issued.

XF: X Flag of CAMAC. True if an X response was generated by the last CAMAC command issued.

OF: TR bit 00. True if bit 0 of the TR is true.

NF: TR bit 15. True if bit 15 of the TR is true.

C12-C15: CTR bits 12-15. True if the corresponding bit of the channel CTR is true. Bits 12-15 of the CTR may be set under program control (e.g., with an instruction like MV, SRC, or CTR with the appropriate bits set in the SCR register).

D. Illustrative Example Programs

1. Initialize MBD and CAMAC. The MBD processor and the CAMAC branch are initialized in single-cycle mode. The Reset MBD function of the CSR takes precedence over all MBD functions and causes it to cease operation at the end of the current MBD cycle. Branch and CNAF commands can be used to initialize the CAMAC controllers and modules. The process is illustrated in Program I. Note that MBD instructions (e.g., MV+PDR+BAR+BC1 = MV PDR, BAR, BC1) are constructed by adding the appropriate mnemonics, which must be defined for the assembler as in Program I, or by defining all the mnemonics of Table IV permanently. Note also the use of @# to generate absolute addresses.

Reset MBD: Loading the CSR with the Reset MBD command (bit 1) halts and resets the MBD processor.

Clear Branch: The BZ (clear and initialize) CAMAC command can be issued to the CAMAC branch by loading and executing a BZ instruction through the IR (in single-cycle mode). CAMAC requires a delay of 15 μ sec before the next branch command.

PROGRAM I

CODE TO INITIALIZE CAMAC

```

CSR=764000      ;CSR (CPU REGISTER)
BZ=7400         ;BZ COM (CTRL)
IR=764006      ;IR (CPU REGISTER)
MSK=764004     ;MSK (CPU REGISTER)
PDX=764002     ;PDX (CPU REGISTER)
MV=0           ;MV (OP)
PDR=340        ;SRC PDR (MBD REGISTER)
BAR=15         ;DST BAR (CAMAC REGISTER)
BC1=3000       ;BC1 (CTRL)
MBDINZ: MOV    #3,@#CSR      ;RESET MBD (CSR FUNCTION BIT 1)
          MOV    #BZ,@#IR    ;ISSUE BZ COMMAND (CLEAR CAMAC)
          MOV    #10,R0      ;WAIT 15 MICROSECONDS AFTER BZ
MBD11:  DEC    R0
          BGT    MBD11
          MOV    @#MASK,@#MSK ;SET SELECTED BITS OF GLR MASK
          MOV    #FNABGN,R0  ;FNABGN = BEGINNING OF CNAF COMMAND LIST
MBD12:  MOV    R0,@#PDX      ;TRANSFER CAMAC SETUP COMMANDS
          MOV    #MV,PDR, BAR,BC1,@#IR ;ISSUE CNAF COMMAND (F8=1)
          CMP    R0,FNAEND    ;CHECK FOR END OF CNAF COMMAND LIST
          BLO   MBD12        ;CONTINUE CAMAC SETUP
          BR    MBDLD        ;GO TO NEXT PROGRAM
MASK:   177400             ;MASK BITS 9-16 OF GLR INTERRUPTS ON
FNABGN: 101751           ;F24 C1 N30 A9 = TURN INHIBIT OFF
          121752           ;F26 C1 N30 A10 = ENABLE BD (GL)
FNAEND:

```

Enable Interrupts: The mask (MSK) for the low-order GL interrupt bits (GLR) may be cleared or selectively set directly by the CPU. A list of CNAF commands to enable or disable interrupts can be issued by loading them through the PDR to the BAR (by loading and executing the IR). Inclusion of a BC1 (or BCO) CTRL command issues the CNAF command in the BAR. The following CNAF commands are generally useful.

```

101751  C1, N30,   A9, F24 = turn inhibit (I) off.
121751  C1, N30,   A9, F26 = turn inhibit (I) on.
101752  C1, N30,   A10, F24 = disable branch demand
                          (BD) for GL.
121752  C1, N30,   A10, F26 = enable branch demand
                          (BD) for GL.

```

2. Load Programs into CM. The programs for the MBD processor must be written in MBD language, but included in the loading PDP-11 computer program. A short program to transfer the proper code from the computer to the MBD is illustrated in Program II. The entire programming for the MBD is assumed to be stored in the computer program between MBDBGN and MBDEND.

Load Program: The program is loaded word by word through the PDX. The ST instruction with an initial address of ADDR = 0 is loaded into the IR to effect the transfer to the location at ADDR in the CM. The pointer (R1) to code in the computer is incremented (1 location = 2 bytes) in synchronization with the address (1 location = 1 word) of the ST instruction (RO).

PROGRAM II

CODE TO LOAD INSTRUCTIONS FROM COMPUTER MEMORY TO CM

```

MBDL0D: MOV    #MBDBGN,R1    ;MBDBGN = BEGINNING OF MBD PROGRAM LIST
          MOV    #ST,R0      ;ST = STORE PDX INTO CM AT ADDR=0
MBD21:  MOV    (R1)+,@#PDX    ;STORE PROGRAM STARTING FROM LOCATION 0
          MOV    R0,@#IR      ;LOAD AND EXECUTE ST INSTRUCTION
          INC    R0           ;POINT ADDR TO NEXT LOCATION IN CM
          CMP    R1,#MBDEND    ;CHECK FOR END OF PROGRAM LIST
          BLO   MBD21        ;CONTINUE STORING
          MOV    #MBDBGN,R1    ;CHECK STORED PROGRAM FOR ERRORS
          MOV    #LD,R0      ;LD = LOAD CM AT ADDR=0 INTO PDX
MBD22:  MOV    R0,@#IR      ;READ PROGRAM IN CM INTO PDX
          CMP    (R1)+,@#PDX    ;COMPARE WITH ORIGINAL IN COMPUTER MEMORY
          BNE   MBDLD        ;LOAD AGAIN IF ERROR FOUND
          INC    R0           ;POINT ADDR TO NEXT LOCATION IN CM
          CMP    R1,#MBDEND    ;CHECK FOR END OF PROGRAM
          BLO   MBD22        ;CONTINUE CHECKING

```

Check Program: The program can be read back through the PDX with an LD instruction in the IR to check for errors.

3. Initialize Selected Channels. Initialization of channels requires coordination of programs in both the computer and the MBD (e.g., Programs III and IV). The computer should supply (for each channel) pointers to the service routine in the CM and to lists of constants. The MBD program should retrieve these pointers and store them appropriately.

PROGRAM III

CPU CODE TO INITIALIZE CHANNELS

```

MBDCHI: MOV    #3,@#CSR      ;RESET PC TO 0
          MOV    #CHNBGN,R1   ;CHNBGN - BEGINNING OF LIST OF POINTERS
          MOV    #4,R0        ;CHANNEL INITIALIZE (BIT 2 OF CSR)
MBD31:  MOV    R1,@#PDX      ;LOAD POINTER TO DATA LIST INTO PDX
          TST   (R1)+         ;EXAMINE FIRST WORD
          BEQ   MBD34         ;OMIT CHANNEL IF ADDRESS IS 0
          MOV    R0,@#CSR      ;INITIALIZE CHANNEL
          TST   (R1)+         ;INCREMENT LIST POINTER
          MOV    #1000,R2      ;WAIT FOR MBD INITIALIZE PROGRAM
MBD32:  TSTB   @#CSR         ;TEST READY BIT 7 OF CSR
          BMI   MBD34         ;CONTINUE INITIALIZATION WHEN READY
          DEC   R2            ;WAIT FOR ABOUT 10 MSEC MAXIMUM
          BGT   MBD32         ;HALT AFTER 10 MSEC
MBD33:  HALT                    ;HALT WITH CHANNEL NUMBER IN R0
          BR    MBDCHI+1      ;BEGIN INITIALIZATION AGAIN
MBD34:  ADD    #400,R0        ;INCREMENT CHANNEL NUMBER
          CMP   R1,#CHNEND    ;CHECK FOR END OF POINTER LIST
          BLO   MBD31        ;CONTINUE WITH NEXT CHANNEL
          MOV   @#CSR,R0      ;CHECK FOR BRANCH OR BUS ERROR
          BIT   #60000,R0     ;BITS 13 AND 14 OF CSR
          BNE   MBD33        ;HALT IF ERROR FOUND
          JMP   MAIN          ;GO TO MAIN PROGRAM
CHNBGN: CH0BGN+1000000        ;BEGINNING IN CM OF CODE FOR CHANNEL 0
          CH0LST              ;BEGINNING IN COMPUTER MEMORY OF LIST
          0,0,0,0,0,0        ;ONE 0 FOR EACH CHANNEL OMITTED
          CH7BGN              ;BEGINNING IN CM OF CODE FOR CHANNEL 7
          CH7LST              ;BEGINNING IN COMPUTER MEMORY OF LIST
CHNEND:

```

PROGRAM IV

MBD CODE TO INITIALIZE CHANNELS

```

0  MBDBGN:  JVC          ;JUMP VIA CTR TO SERVICE ROUTINE
1          MV,PDP        ;POINTER TO LIST IS STORED IN PDR
3          AD,0,MAR,RDR  ;READ FIRST WORD FROM LIST
4          BCT,*,DCB     ;WAIT FOR UNIBUS
5          MV,MDR,CTR    ;DEPOSIT FIRST WORD INTO CTR
6          INM,MAR,MAR,RDR ;READ SECOND WORD FROM LIST
7          BCT,*,DCB     ;WAIT FOR UNIBUS
10         MV,MDR        ;READ SECOND WORD INTO TR
11         SRR 1         ;CHANGE POINTER FROM BYTE TO WORD POINTER
12         AD,0,ILR      ;DEPOSIT SECOND WORD INTO ILR
13         BCF,*,C15     ;CTR BIT 15 SETS PRL BY EXIT 1
14         EX1           ;EXIT 1 - SET PRL AND EXIT
15         EX2           ;EXIT 2 - RESET PRL, SET CEL, AND EXIT

```

CPU Program: The CPU program is illustrated in Program III. A list of pointers begins at CHNBGN, with two pointers per valid channel and one 0 for each channel that is not to be initialized. The first word points to the service routine in the CM, and the second points to a list of constants stored elsewhere in the computer program. A pointer to the appropriate pair of pointers is loaded into the PDX. Loading the CSR with bit 2 and the channel number in bits 8-10 begins initialization. Bit 7 of the CSR indicates when the channel is accepted and initialization of that channel has begun in Program IV in the MBD. A time-out loop checks for channel malfunction (lack of an EXIT from the previous channel program). While one channel is being initialized by Program IV, the next is being readied in Program III. After initialization, the CSR (bits 13-14) may be checked for errors.

MBD Program: The MBD program is illustrated in Program IV. The code begins at location 0 with a JVC instruction to link interrupts to their own service routines. Initialization begins at location 1, with a (byte) pointer in the PDX to constants in the computer memory. Afterward, the PDX should not be accessed by the MBD because the CPU may change it. Two words are read in through the MDR from the (word) addresses specified by the MAR, and are deposited in the CTR and ILR for later use. Bit 15 of the CTR is checked to see whether to continue initialization and/or execution via an EX1 command. Note that the MBD addresses begin at location 0, whereas the PDP-11 addresses are relocatable. The BCT instructions, for example, branch to their own addresses (location 4 or 7) to execute wait loops. Note also that MBD addresses, including the MAR, are *word* addresses, whereas the PDP-11 code produces *byte* addresses.

4. Data Acquisition. A minimal program to read data from CAMAC registers and store them in computer memory is illustrated in Program V. A block of data is read

PROGRAM V

MBD CODE TO TRANSFER DATA FROM CAMAC TO COMPUTER MEMORY

```

CH7BGN = CH0BGN+100      ;BEGINNING OF CHANNEL 7 TRANSFER
*= MBDBGN+CH7BGN+CH7BGN ;CPU PC IS BYTE POINTER
CH7BGN:  DEM  ILR,MAR,RDH ;POINTER TO BUFFER IS STORED IN ILR
        LCI  CH7LST      ;POINT CM TO LIST OF CONSTANTS
        MV  MEM,WCR      ;RETRIEVE WORD COUNT FROM LIST
        INM CTR,CTR      ;POINT CM TO CAMAC CNAF COMMANDS
        BCT *,DCB        ;WAIT FOR UNIBUS
        JP  CH7RD        ;START TRANSFER WITH CAMAC COMMAND
LOOP7:   BCT *,DCB        ;WAIT FOR UNIBUS
        INM MAR,MAR      ;POINT TO NEXT LOCATION IN BUFFER
        BCT *,BRB        ;WAIT FOR CAMAC BRANCH
        MV  BDL,MDR,WTH  ;WRITE WORD INTO COMPUTER MEMORY
CH7RD:   MV  MEM,BAR,BC0  ;READ NEXT WORD FROM CAMAC
        INM CTR,CTR      ;CNAF COMMANDS IN CM LIST
        DEM  WCR          ;CHECK FOR END OF WORD COUNT
        BCF  LOOP7,ZF     ;CONTINUE WITH NEXT WORD
CH7SET:  INT              ;INTERRUPT COMPUTER CPU
        INM MAR,ILR      ;STORE NEW BUFFER POINTER IN ILR
        LCI  CH7BGN      ;POINT INTERRUPT TO BEGINNING
        BCT  *,DCB        ;WAIT FOR UNIBUS
        EX2              ;EXIT AND RELEASE UNIBUS
CH7LST:  25              ;WORD COUNT (21 CNAF COMMANDS)
        1020             ;FIRST CNAF COMMAND (F0CINIA0)
        ----             ;REST OF LIST OF CNAF COMMANDS
MBDEND:

```

from the CAMAC registers by executing a stored list of CAMAC CNAF commands, and the data are transferred simultaneously to the computer memory. The registers may be read in any order, as specified by the stored list of CNAF commands. A pointer to the buffer in the computer memory is assumed to be stored in the ILR. The procedure of Programs III and IV may be used to load the beginning address into the ILR initially, and to reset the pointer to the beginning of the buffer periodically. The value of the pointer left in the ILR at the end of the transfer is the location of the next free address. An execute INT instruction at the end of the transfer alerts the computer CPU to the presence of a new block of data in the buffer of the computer memory.

The block transfer is effected in the loop of instructions between LOOP7 and CH7SET. Note that the write instruction holds the Unibus to minimize waiting time (the first read instruction is a dummy instruction to acquire and hold the Unibus). For most combinations of MBD and computer memory, the timing is such that the BCT instructions in this loop may be omitted so that the time for transfer is only six cycles of the MBD per word. If it is inadvisable to use the hold mode for data-channel transfers (because of disk manipulations, etc.), the WTH instruction can be replaced with a WTR instruction, in which case the BCT+DCB instruction must be retained and the RDH instruction must be removed. The EX2 instruction at the end releases the hold on the Unibus as well as relinquishing control to other channels until the next interrupt on channel 7.

5. **Stand-Alone Bootstrap Loader.** Two possible modes of MBD operation, stand-alone and remote branch driver, require a way of loading the MBD control memory. Peripherals can be interfaced either to the Unibus or via CAMAC. Efficient Unibus use requires some MBD modifications. For LAMPF application, the interface will be in CAMAC and will generally be a teletype or high-speed reader. For remote operation, the control memory could be loaded from the high-speed data link.

Sample Program VI is a bootstrap loader for an ASR 35 paper-tape reader. The bootstrap has 27 instructions that must be manually loaded into the MBD. This can be done with only a control console, but it takes some time and requires several step sequences. If the MBD has the front-panel switch option, all that is required is to load PC counters with the desired address minus one and then load the instructions in the front-panel switch and press the transfer switch on the control console. The following sequence should be followed if the MBD has front-panel option.

- a. Set single-cycle mode and manual mode.
- b. Set PC counter with JP to address minus one.
- c. Load instruction to be stored in switches on MBD.
- d. Using control console, load IR with MV, SWS, and MEM. and press transfer switch.

PROGRAM VI
MBD BOOTSTRAP LOADER FOR ASR-35 PTR

0	JVC		120000
1	MV DAR,BAR,BC1		003055
2	BCT *,BRB	;CLEAR INHIBIT	101002
3	MV CCR,BAR,BC1		003115
4	BCT *,BRB	;START READER	101004
5	A1: MV PC,ILR		000201
6	JP RDWRD	;GET STORE INST	170014
7	ST Ø,INST	;PLANT STORE INST	130012
10	MV PC,ILR		000201
11	JP RDWRD	;GET DATA WORD	170014
12	INST: RSWD1	;SPACE FOR PLANTED INST	
13	JP A1	;NOT DONE YET	170005
14	RDWRD: LCI A2		160016
15	JP RDBYT	;GET LO BYTE	170026
16	A2: MV BDL,WCR		000263
17	LCI A3		160021
20	JP RDBYT	;GET HI BYTE	170026
21	A3: INM ILR,CTR	;BUMP TO RETURN ADDRESS	010025
22	MV BDL,Ø		000260
23	SLL 8	;PACK WORD	154010
24	IOR WCR,Ø		050060
25	JVC		120000
26	RDBYT: MV GP1,BAR,BC1		003155
27	BCT *,BRB	;TEST TTY STATUS	101027
30	BCF RDBYT,QF	;TEST Q FOR STATUS	114026
31	MV MDR,BAR,BCØ		002635
32	BCT *,BRB	;GET BYTE	101032
33	JVC	;RETURN	120000
	END		

LOAD the specified file register with the following CNAF command.

```

C1 F24 N30 A9 in DAR ;CLEAR INHIBIT
C1 F9 N1 A0 in CCR ;START READER
C1 F27 N1 A1 in GP1 ;TEST STATUS
C1 F2 N1 A0 in MDR ;READ BYTE

```

Repeat c and d until control program is loaded. The DAR, CCR, GP1, and MDR must be loaded with the CNAF command shown in the example. With the boot, any control program can be loaded from paper tape. The program is loaded at location 0-33, but it could be anywhere in the control memory. The program code is shown and can be easily modified. The program starts the reader, collects two bytes and packs them into one 16-bit word, then stores the word and increments the address for the following word. A "Q" list indicates the teletype status.

As a stand-alone device, the MBD must be able to change channels and initiate a channel operation. This is accomplished by loading the CIL register with the desired channel number. The CIL is loaded by executing a move, any source, with the desired channel to a "O" destination and a CI control command. Then on the next exit command, the number set in to CIL will be in the priority structure of the MBD and will be accepted to run when it is the highest priority request. If the MBD is used as a stand-alone or remote device often enough to warrant implementing the loading operation automatically, a hardware boot will be added to it.

IV. MBD PERFORMANCE SPECIFICATIONS

The MBD discussed here has been prototyped and checked out and is in operation with the following characteristics and capabilities. Nine additional units are being built and are in various stages of assembly and testing. Many more units will be required as the device establishes itself as a stand-alone branch driver.

The prototype MBD has 436 IC's and required 4352 wires to assemble. The unit was wired in 30 h using the CASH machine-aided wiring technique. The cost is within the design goals. The Laboratory costs are as follows.

Integrated Circuits	\$1975
Hardware (Including Power Supply)	1925
	\$3900
Labor (Assembly and Checkout)	400
Cost Each (Not Including Design Costs)	\$4300
Design Cost (\$30,000 Prorated for 10 Units)	3000
TOTAL MBD COST	\$7300

The first 10 units have 1024 words of memory and pay the engineering cost and are still within the design range. Additional units with only 256 words of memory will cost approximately \$3500 each.

The following is a summary of MBD specifications.

Options

Memory Size 256 to 1024 16-bit words in 256-word increments using the 256x1-bit memory device. Using the

1024x1-bit device, the memory size is 1024 to 4096 words in 1024-word increments.

Front-Panel Options include lights for the program counter and the D bus and switch inputs to the S bus. The switch inputs are active only in the manual mode of operation.

Manual Controller is required only if the MBD is to be used as a stand-alone branch driver. The controller is a PDP-11 simulator connected to the MBD by the standard Unibus cable. The unit has its own internal power supply.

Power Supply is a separate 5-V dc, 25-A logic power supply.

Operating Speed

The MBD has a 350-nsec instruction cycle time. In the *link-list mode* of operation, a 16-bit word can be transferred from CAMAC module to PDP-11 memory in 10 μ sec. If the *CAMAC command list* is in the MBD control memory, the transfer time for 16-bit words is 5 μ sec. In the *block-transfer mode*, the transfers are made at the speed of the PDP-11 memory. The maximum speed of CAMAC branch operation is one 24-bit word per microsecond.

Software

An extensive set of *diagnostic programs* has been written to test the MBD. These include tests of memory, file register, instruction set, DMA transfers, interrupts, CAMAC branch, and the various modes of operation. With the aid of a test CAMAC module, a complete system test can be made. A *macro assembler* has been developed for the MBD using the PDP-11 computer. Using the mnemonics discussed here, the assembler sums the octal representation of the fields to form the instruction. Many operational programs have been written, including all of the modes of MBD operation, control memory load programs from the PDP-11 and bootstrap loaders from CAMAC peripherals, and MBD and CAMAC initialize programs.

Mechanical

The MBD is housed in a standard drawer that is covered and has two cooling fans. The drawer is 3-1/2 x 19 in. and has overall depth of 24 in. with the standard Unibus connector in the rear. The unit weighs approximately 20 lb. The CAMAC connector is on the front panel and is

the standard 132-pin Hughes connector. The CASH system has a variety of cards using 14-, 16-, and 24-pin sockets for integrated circuits, and the wiring has a two-level wire wrap using No. 30 wire.

Temperature

The MBD has a power dissipation of approximately 100 W. The unit has two cooling fans that circulate ambient air. The IC's used in the MBD operate at 0 to 70°C.

ACKNOWLEDGMENTS

We are indebted to Richard F. Thomas, Jr. and Donald R. Machen of the Los Alamos Scientific Laboratory, who greatly influenced the MBD systems design and operation, and to Kenneth A. Klare for the many programs used in testing the MBD.

REFERENCES

1. H. L. Anderson, P. R. Bevington, J. A. Buchanan, H. S. Butler, et al., "LAMPF Data Acquisition System," Los Alamos Scientific Laboratory report LA-4504-MS (1970).
2. "A Modular Instrumentation System for Data Handling," EURATOM 4100e. Available from L. Costrell, National Bureau of Standards, Washington, D. C.
3. "Organization of Multicrate Systems," EURATOM 46003, *Ibid*.
4. Digital Equipment Corporation PDP-11 Unibus Interface Manual, DEC-11-H1AA-D.
5. Digital Equipment Corporation PDP-11 Handbook 112X01269, AJOF1150.
6. "Design of Micropogrammable Systems," Application Notes, Signetics Memory Systems, Sunnyvale, California, December 1970.
7. "Economic Advantages of Microprogramming," Application Notes, Signetics Memory Systems, January 1971.
8. Gary D. Hornbuckle and Enrico I. Ancona, "The LX-1 Microprocessor and its Application to Read-Time Signal Processing," IEEE Trans on Computers, 6-19, No. 8 (August 1970).
9. Samir S. Husson, *Microprogramming Principle and Practices*, Prentice-Hall, Inc. 1970.
10. "MSI/TTL Integrated Circuits from Texas Instruments," Bulletin CB-125, Texas Instruments, Dallas, Texas, 1970.
11. *Designing with TTL Integrated Circuits*, Texas Instruments, Inc., 1971.

KT:763(520)