To be published in IEEE Trans. on Nuclear Science

LBL-1577
Preprint c.1

SOME CHARACTERISTICS OF INTERFACES BETWEEN CAMAC AND SMALL COMPUTERS

Frederick A. Kirsten

February 1973

Prepared for the U.S. Atomic Energy Commission
under Contract W-7405-ENG-48

DISCLAIMER

SOME CHARACTERISTICS OF INTERFACES BETWEEN CAMAC AND SMALL COMPUTERS*

Frederick A. Kirsten
Lawrence Berkeley Laboratory
Berkeley, California 94720

Summary

The typical CAMAC system is operated in conjunction with a small computer. In the usual case, the computer acts as a repository for data generated by CAMAC, and also controls and directs the CAMAC operations. This paper discusses some of the aspects of the hardware interface, and also some of the interaction between the computer and the CAMAC system.

Introduction

CAMAC systems can be assembled in many shapes, sizes, and configurations. The flexibility of CAMAC, which permits the accommodation of a wide range of problems, is one of its greatest assets. There exist, in actuality or on paper, systems of all sizes, from those completely self-controlled and contained in a single crate, to multibranch, multi-computer systems involving a very high level of data interchange and communication.

This paper concentrates on a specific, relatively simple type of CAMAC system--namely, one whose crates are interconnected by a single branch highway, and controlled by a single source of system control--a computer. However, reference is also given to systems in which individual crates are interfaced directly to the computer. The discussion centers around the problems of interaction and intercommunication between computer and the CAMAC system, and the hardware necessary to effect the interfacing.

It is assumed that the reader is familiar with both the CAMAC Crate Dataway, The CAMAC Branch Highway, and the Crate Controller Type A. These are described in the specifications[1,2] and in other papers of this series.[3-6] In addition, familiarity with the basic facilities of a typical small computer is assumed.

The "Typical" Small Computer[7]

The computer interacts with CAMAC via what is often called its "I/O structure." This term refers to capabilities of the computer to communicate with devices external to itself. (I/O is the abbreviation for Input/Output.) For the purposes of discussion, the "typical" small computer is assumed to have the following characteristics in its I/O structure.

(a) A programmed I/O facility. This refers to the ability to transfer data between computer and external device under the complete step-by-step control of the computer program. In general, the transfer of each word, or the emission of each I/O command, is the result of one or more instructions in the program.

(b) A block transfer facility: This permits transfers of blocks (large numbers of data words) in such a way that the computer program is required to issue only the initial instructions for the block. The remainder of the operation proceeds under hardware control.

*Work performed under the auspices of the U.S. Atomic Energy Commission.

(c) An interrupt facility, which permits the external device (CAMAC) to gain the attention of the computer program.

(d) An I/O command repertoire, usually relatively limited, by which specific I/O operations can be commanded.

Comparison of Crate Controller and Branch Driver Interfacing

The interface between a computer and a CAMAC system is usually placed either at the crate controllers of the individual crates, or at the branch driver of a branch highway system. Figure 1 shows an example of each configuration. The prime purpose of either of the two configurations is to provide a vehicle for the conversations between computer and modules. The main difference between the two is in the organization of the conversation and paths. The branch system in Fig. 1a provides a single port through which the computer "talks" to the entire system. The other system, shown in Fig. 1b, has an interface at each crate. Let us call it a "radial" system, since the communication radiates from the single computer to the several crate controllers. For convenience we refer to the crate controllers of this type[8] as Type U.

A given collection of CAMAC modules can often be interfaced either way. Both ways are applicable to single-crate and multicrate systems. Which method is better depends on details of the particular application. Some bases for comparison are given below.

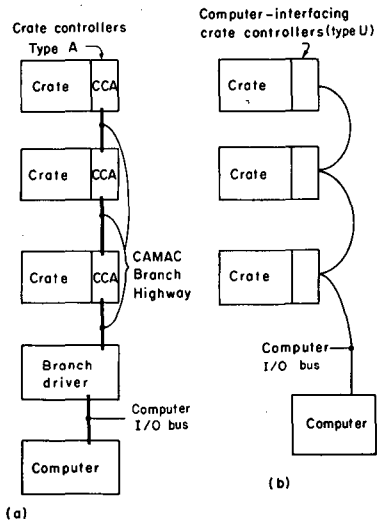

Fig. 1. Two examples of how a three-crate CAMAC system may be configured: (a) using a CAMAC Branch Highway interconnection; and (b) using a computer I/O bus to interconnect the crates.

Cost. The three interfacing components shown in Figs. 1a and 1b are the branch driver, the Crate Controller Type A (CCA), and the computer-interfacing Type U Crate Controller. Based on an estimate of the amount of logic required in each of the three, the branch driver will probably cost the most, the CCA the least, and the Type U somewhere in between. To implement a branch system, one branch driver and a CCA for each crate are needed. The radial system requires one Type U Crate Controller for each crate. Either the radial or the branch system may have the lower total component cost, depending on the number of crates in the system.

Within a laboratory, the long-range cost of a set of CAMAC systems can be influenced by the amount of trading of components among the systems. For a system that never changes, one may consider only the initial cost. If systems are continually being broken apart and reconfigured, the initial cost may not be the proper criterion. For example, to change the model of computer used on a branch highway system, one needs to change only the branch driver, whereas, for a radial configuration, all Type U Crate Controllers have to be changed.

Operational characteristics. With the branch highway scheme, intercrate communication can be accomplished completely outside the computer structure. This sometimes has benefits. For example, block transfers of data involving modules from several crates can probably be handled more easily with the branch highway--the crate boundaries can be made transparent to the computer interface. On the other hand, simultaneous block transfers to two or more separate modules, using, e.g., interleaved cycles on two separate data channels, can possibly be done more easily in the radial system, particularly if the modules are in separate crates.

Type U controllers usually can provide a closer coupling between the Look-at-Me's generated by the modules, and the interrupt structure of the computer than a branch coupler.[13] In a radial system, it is generally the case that the proper computer subprogram can be entered in less elapsed time and with fewer operations.

CAMAC-Computer Communication

Certain common problems arise whenever two data structures are interfaced. This is no less true if one of the data structures is CAMAC. In the following, we

TABLE I

Comparison of characteristics of CAMAC and I/O structures of typical small computers

Topic	CAMAC characteristics	Characteristics of the I/O structure of a typical small computer
Types of operations that may be carried out	Bidirectional data transfer	Bidirectional data transfer [two modes: Programmed I/O (accumulator transfers), autonomous transfers (e.g., data channel)]
	Commands directed to modules [A much larger command repertoire than in the average computer's I/O structure]	Commands directed to peripheral devices [different command structure from CAMAC]
	Look-at-Me service requests	Interrupts or Data Channel cycle requests
Basic cycle time	Cycle time $\geq$ 1 μsec	Cycle time varies from one computer to next, typically of the order of 1 μsec
Timing and logical control of operations	Specified by CAMAC specifications	Different for each computer (and different from CAMAC)
Data word size	Maximum word size = 24 bits. Actual word size (no. of bits that are actually significant) depends on the module involved.	Word size is generally less than 24 bits
Service requests/interrupts	At the Crate Controller, there are available up to 23 bits of service-request information. At the Branch Driver, there is 1 bit (the BD signal) available at all times; after a GL cycle, there are 24 bits available. More can be obtained by addressed commands	Service requests include: Interrupts: single-level or multiple levels Data Channel cycle requests: possibly on more than one channel; generally, transfer can be made to or from computer Direct memory access cycle request. The first can be spontaneous; the latter two must usually be "set up" by a preceding routine

discuss how some of these common problems relate to the CAMAC-computer interface.

Table I compares some characteristics of CAMAC and the I/O structures of typical small computers. The table substantiates that translations are necessary--from the I/O "language" of the computer to the CAMAC "language," or vice versa. The entities to be translated include the size of data words, commands (i.e., the significance of commands), and timing sequences.

Timing problems are often solved by "staticizing" the data and commands. This is particularly true if the I/O structure uses a synchronous transfer--i.e., it is timed by a computer clock, and cannot wait for CAMAC. This is done by providing registers accessible from either side of the interface. (A register is a one data-word or one command-word storage device, probably composed of one flip-flop for each bit involved.) The registers behave as "mailboxes." One entity, CAMAC or computer, deposits information in the registers, using its own cycle timing and sequencing. The other entity, using its own cycle timing quirks, can then pick up the information. One entity may completely finish an operation before the other begins, thus removing the necessity for interlocking and synchronizing two different sets of cycle timing characteristics.

If the I/O structure uses an asynchronous cycle--i.e., one that can be momentarily pause to wait for CAMAC, it may be possible to imbed the CAMAC cycle within the I/O cycle. If so, the "mailbox" technique may not be used. Examples are given in a reference.[10,13]

In addition to the registers, an appropriately designed control sequencer and organizer coordinates the interlocking of the complete operations of CAMAC and of the computer.

A Typical Interface

Figures 2 and 3 are simplified block diagrams showing the basic parts of an example of a CAMAC-computer interface. Figure 2 shows the parts primarily concerned with the transmission of data; Fig. 3 shows portions concerned with the transmission of CAMAC commands and control features.

It should be emphasized that this is an example. Some of the facilities described below will obviously need to be in all branch drivers. Other facilities or features can be included at the designer's option.[9] The user therefore cannot assume that all branch drivers contain all features. For example, some perform programmed I/O transfers only. Some have a very rudimentary means of handling requests.

Data-Oriented Parts

Figure 2 illustrates the flow of data to and from the branch highway bidirectional Read/Write (BRW) bus, and to and from the computer. Some data are transferred between computer and interface via programmed I/O (accumulator) transfers under complete software control. This means every step of the process is specially controlled by the computer program; the interface has a relatively simple task. Other data are transferred in block transfer mode [shown here as Data Channel (DCH) transfer.] In this mode, the computer program initializes the system, but each individual data word in the block is transferred under autonomous (hardware) control. Depending on the computer, more or less of this hardware must be in the interface.

Data Registers. Figure 2 shows two data registers, one for Programmed Input-Output (Prog I/O) transfers

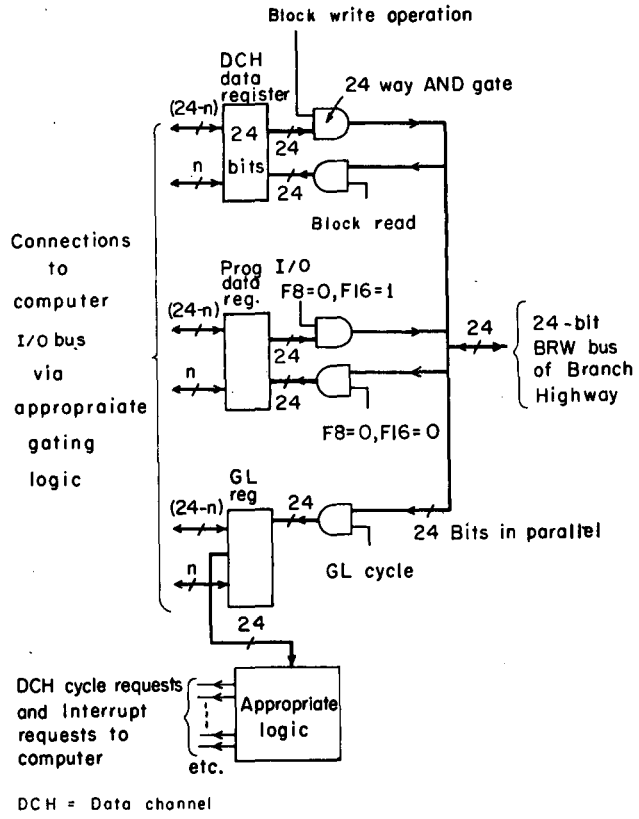

Fig. 2. A block diagram showing some of the parts of a typical computer-branch highway interface that are concerned with the data bus (BRW) lines of the highway. It is assumed that the computer has a word size of N bits.

and one for Data Channel (DCH) transfers. Two registers are necessary if a DCH cycle is to be interleaved with steps of a Prog I/O transfer. Each is a 24-bit register in order to accommodate the 24-bit CAMAC data word. Typical small computers have word sizes of 12, 16 or 18 bits. This makes it necessary to perform two computer I/O transfer cycles to move the full 24 bits of data from the data register to computer memory. The first transfer generally moves bits 1 through n, where n is the computer word size; the second then moves bits (n+1) through 24. However, there are many cases where not all 24 bits are significant. As an example, consider a 16-bit computer reading a module which contains a 16-bit scaler. The CAMAC operation will move 24 bits from module to register. However, since only 16 bits are useful, and since here n = 16, only one I/O transfer need be made.

GL register. A third register, involved only in data flow from CAMAC to computer is the GL register. During the Graded-L cycle, a 24-bit word carrying information on the status of L requests in the branch is generated on the BRW bus. The GL register saves this information. It may be made available to the computer via Prog I/O transfer. In some branch driver designs, certain GL bits can be assigned to automatically cause computer interrupts. In some, an interrupt vector is included, which directs the computer to a service subprogram that is appropriate for the particular CAMAC service request.[13] Another option is to use certain GL bits to synchronize block transfers.[12] Whenever these bits come up, another word is transferred.

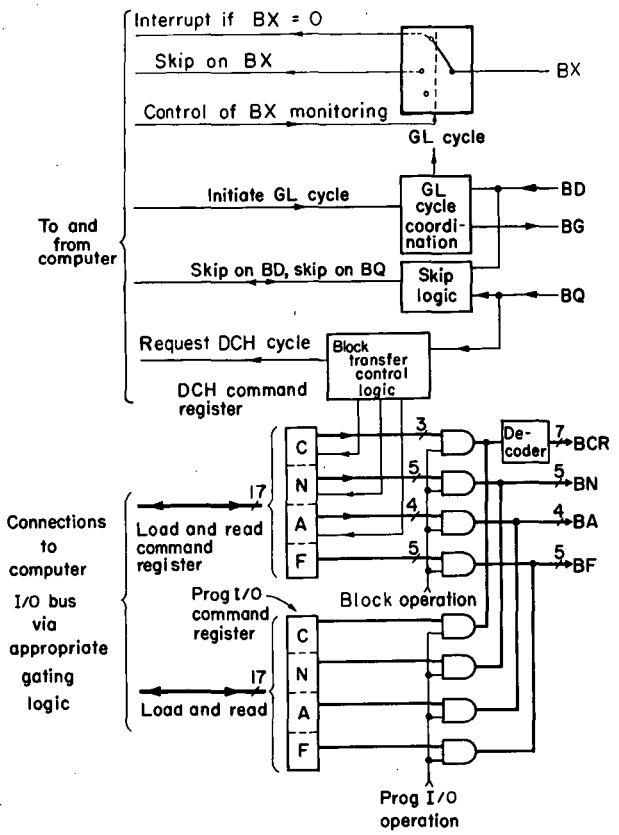

Fig. 3. A simplified block diagram showing the parts of a typical computer - branch highway interface that are concerned with branch highway commands.

Command-Oriented Parts

Figure 3 shows the parts of the typical interface that are concerned with issuing addressed CAMAC commands, with the BQ signals, and with the Look-at-Me servicing.

Command registers. As with the data, separate command registers are shown for Prog I/O and DCH transfers in order to permit interleaved Prog I/O and DCH cycles. In most cases, the CAMAC commands are generated by the computer software, and are transmitted by the computer to the command register in a form the computer considers as data. Thus, the loading by the computer of the command register is similar to loading the data register.

To put it another way, the command repertoire of the average computer I/O structure is limited, and is much smaller than the CAMAC command repertoire. Thus, there is often no way a direct translation from I/O command to CAMAC command can be accomplished. See the references[10,13] for examples of exceptions to this statement.

A complete branch highway command requires 17 bits--3 bits for Crate address, 5 for Station Number, 4 for Subaddress, and 5 for Function Code. Thus, double word transfers to load the Command Register may be required for computer word sizes of 12 or 16 bits. Commands addressed to more than one crate or more than one station require special handling.

Q responses. Certain CAMAC commands require that the module give a Q response. On the branch highway all Q responses from individual crates are gathered onto the BQ line. The state of the BQ line may be strobed into a one-bit BQ register during each Prog I/O-initiated CAMAC cycle and made available to the computer program. The program may test this register and may conditionally branch, depending on its state. A second one-bit BQ register may be required for Q-controlled block transfers.

Operational Sequences

In this section, the sequences of events in several types of operations are briefly explained, with emphasis placed on the interplay between the computer and the CAMAC system.

Programmed I/O Data Transfers

In these operations the object is to move data between (the accumulator of) the computer and a specific CAMAC address (e.g., a register in a module). The computer program is in control in that it specifies the direction of data flow, the CAMAC address, and when the operation is to take place. Two examples are given in Table II. One is for a Write operation in which data move from computer to module. The other is for a Read operation, with data flow in the opposite direction.

Block Transfer Operations

In block transfer operations, the object is to move blocks (many words) of data with the minimum intervention on the part of the computer program. Whereas the initiation of the block is under program control, the transfer of individual data words is controlled by relatively simple hardware logic.* Generally, the same CAMAC function code--e.g., F(0)--is used during the entire block.

The CAMAC specifications[1] define three modes of block transfer control using Q-responses. These are called: Address Scan, Stop and Repeat modes. The Address Scan mode is used to move data to or from a group of registers that reside in an array of modules. In this mode, a Q-response of '1' indicates that the register addressed by the current command actually exists. A response of '0' says, "There's no (scan-addressable) register at this location." These responses enable the Branch Driver to transfer data only to (or from) those CAMAC addresses (CNAs) that contain registers.

The Repeat mode is used to move data to or from a single register. It is intended for situations where the register may not always be ready for the transfer. In this mode, the Branch Driver may repeatedly try to effect the transfer. A Q response of '1' from the module says the transfer was successfully accomplished (as far as the module could tell). A response of '0' means "I wasn't ready, please try again." The Repeat mode is therefore one of several methods available for synchronizing the readiness of computer and module.

The Stop mode is also used to move data to or from a single register. It is intended for situations where the number of words in the block is not initially known, and where the module is able to determine when the last

*This statement refers to this example. The algorithm described can also be executed under complete software control, using a series of Prog I/O data transfers.

TABLE II

Examples of Sequences for Programmed I/O Data Transfers

	Computer I/O operation	CAMAC operation
I.	WRITE--(Data flow from computer to CAMAC address)	
	A. Load data into the "Prog I/O data reg" in interface. (This may be a double transfer, if data word size > computer word size.)	IDLE
	B. Load CAMAC command into "Prog I/O command reg" in interface. (This may also be a double word transfer.)	IDLE
	↓	
	Wait for signal that Branch Highway cycle is done (e.g., Program loop using "Skip if BH cycle done").	C. Execute Branch Highway cycle, transferring data loaded in A using CAMAC address loaded in B.
	↓	
	D. Do next sequence - - - -	
II.	READ (Data flow from CAMAC address to computer.)	
	A. Load CAMAC command into "Prog I/O command reg."	IDLE
	↓	
	Wait for signal that Branch Highway cycle is done.	B. Execute Branch Highway cycle, transferring data from CAMAC address given in A to "Prog I/O data reg."
	↓	
	C. Examine BQ register if Q response defined for command used. BQ register may be tested by program using, e.g., "Skip if BQ = 1." Also check BX.	IDLE
	D. If BX "1, and if proper BQ response detected, transfer data from "Prog I/O data reg." to computer.	
	E. Do next sequence.	

word of the block has been moved. A Q-response of '1' indicates the word transferred is part of the block. The first time a '0' is received, it indicates the previous transfer was the last word of the block.

Many branch driver designs include the hardware by which block transfers are coordinated. Inputs to this hardware segment include: 1) instructions from the program telling which mode of block transfer to use; and, 2) the BQ signal. Outputs from the segment include: 1) the decision whether the transfer of a data word has been effected; and 2) in the case of Scan Address mode, CAMAC address updating signals, according to an algorithm described elsewhere.[3]

It is good practice to include in the interface a means of insuring that the block transfer will always be terminated within some safe limits. In the case Repeat or Stop, this may be by specifying a maximum number of words to be transferred (Word Count). In Stop mode, then, the block would be terminated either by hitting the Word Count limit or by a Q=0 response.

For Scan Address, the limit could be specified either by Word Count or by the last CAMAC address which is to be scanned. **Preferably both are included.**

Table III shows an example of the sequences of operations in executing the Stop mode of block transfers.

Since the computer program is not involved with the block after step C in the sequence of Table III, it must somehow be notified when the block is completed. This can be done by the interface either actively or passively. An active notification is accomplished by interrupting the computer. The sub-program that services the interrupt takes any appropriate action. Passive notification may consist of setting a "Task done" flag. The program can test this flag at its leisure.

TABLE III

Example of sequences for a Q-controlled block transfer of data via data channel. The example depicts a Stop mode Read transfer, in which data move from modules to computer memory.

	Computer I/O operation	Operation within interface	CAMAC operation
I.	Set up operation with Prog I/O transfers:		IDLE
	A. Load appropriate CAMAC commands into "DCH command reg." Command is interpreted by interface to learn direction of data transfer--e.g., F(0), Read Group 1 Register, is transfer from module to computer.		
	B. Load "mode of transfer" into block transfer control register of interface. Mode is Scan Address, Repeat or Stop.		IDLE
	C. Load Word count control register.		
II.	The transfer itself: Once the setup is done, data words are transferred, one per cycle, until the block transfer is done, All this is done under hardware control. The computer program is off doing something else.		
			D. Branch highway cycle fetches data word from the module and delivers to interface.
		E. Examine the BQ register. If BQ = 1, do step F. if BQ = 0, go directly to step H.	
	F. Data channel cycle sends data word to computer memory.		IDLE
		G. If number of words transferred = Word count, go to step H. If less than Word count, go again to step D.	
III.	When the transfer is done, the computer program is notified of that fact.		
	H. Interrupt the computer.		

TABLE IV

Examples of L requests and appropriate responses

Type of module	Meaning of request	Proper service (besides resetting the L-request flag
(a) Analog-to-Digital Converter	A data word is ready for transfer	Transfer the data
(b) Scaler	Scaler has overflowed	Increment a computer memory location that is used to count overflows.
(c) Typewriter control	Typing of previously received character is completed	Send another character if there is one
(d) Magnetic tape Controller	Previously received word has been recorded on tape	Initiate data-channel cycle to send another word
(e) Arbitrary module	No significance in this particular system	Ignore it

Responses to Look-at-Me's

Modules have the facility to request service by means of the Look-at-Me signal. Different modules have different reasons for requesting service; they request different types of service; and the requests from different modules carry differing degrees of urgency. The computer and (or) computer interface have the duty of interpreting the requests and of initiating the proper action.

To make this more explicit, let us examine some cases. Table IV lists some of the reasons why a module might generate an L request, and what would be the appropriate service.

The problem involved in matching the action to the request exists both in the crate controller-to-computer type of interface and in the branch driver-to-computer interface. Let us concentrate on the latter, which is somewhat more complex.

The Branch Demand (BD) signal of the branch highway is the summation of all service requests from all crates. By itself, it carries so little information that it almost inevitably results in further action. Figure 4 shows graphically the tree-like structure that can be imagined as representing the L-source identification process. The BD signal carries one bit of information. If there is more than one source of L requests, a GL cycle is executed. This immediately makes 24 bits of information available. On the basis of the GL word, the computer program must decide what computer action is required either to service the request, or, if insufficient data are in the GL word, to further identify the source of the L request. If there are 24 or fewer sources of L request, the search can stop here; the requesting module can probably be identified at this point. If there are more than 24 sources, then the GL word indicates the direction the software search should take in order to reduce the searching time.

Notifying the computer. As described above, there are certain tasks involved in identifying the source of an L request. In a particular interface, a given task may be allocated to hardware or software. To start at the bottom of the tree in Fig. 4, the computer can be notified of the existence of a BD in two ways. The 'active' way is to request an interrupt. This gains the attention of the computer within a few microseconds. The passive way is to set a flag, and wait for it to be tested when the program gets around to it. The delays involved depend on how pre-occupied the program is. The action taken by the program when it becomes aware of the BD is to instruct the interface to do a GL cycle. Following this, the program will interpret the GL word and initiate the proper service.

An alternative procedure for the interface is not to tell the computer about the BD, but rather to take a step of its own. This step is to do a BG (graded-L) operation. Having gotten the GL word, the interface has two choices. The first choice is to request an interrupt. (If we've gone this far, flags are too slow.) The purpose of this interrupt is to notify the computer that the GL-word is available.

The second choice available to the interface is to interpret the GL-word itself. As a result of this interpretation, the interface could be designed to do the following:

(i) interrupt the computer on one or more levels of priority;

(ii) request a data channel cycle to transfer data into the computer memory;

(iii) request a data channel cycle to transfer data from the computer memory;

(iv) execute an "increment contents of memory" cycle at a specified memory address.

Some Branch Driver interfaces[11,12] have been designed with an internal memory and small control processor. This enables them to execute fairly complex algorithms, autonomously. With this type of interface in mind, the list is ended:

(v) without bothering the computer at all, issue the necessary CAMAC commands to service the L-request, and to reset it.

A Sequence. A possible sequence of events following the initiation of a typical L request follows. This sequence is written after assuming certain choices in the design of the interface. The choices should be obvious.

(a) Module X raises its L request(L(x)=1).

(b) This results in the Branch Demand going to the true state (BD=1).

(c) The hardware of the branch driver recognizes the BD, and executes a GL cycle as a result. The GL word appears in the GL register. Because of action (a), a certain bit in the GL word, bit Z, will be '1'.

(d) The Branch Driver recognizes that bit Z is in the 1 state. Having been prewired to do so, it initiates the proper computer action. Let us imagine that this action is to interrupt the computer on priority level Y, setting into motion a service program.

(e) The nature of the specific L request now being fully recognized, the service program does the appropriate servicing that the L request had originally asked for.

(f) During the service program, the L flag in module X is reset, perhaps with an F(10)--Clear Look-at-Me--command. (If the module is so designed, the L flag may have automatically been reset at the instant the servicing was done.)

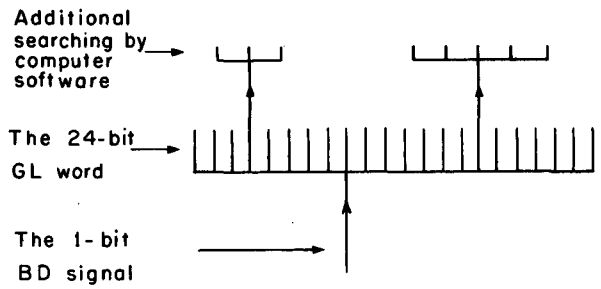

Fig. 4. This tree-like structure illustrates the possible steps in identifying the source of an L request.

(g) After L from module X is reset, BD returns to 0 unless another L from another module has appeared in the meantime.

Command Accepted, BX

The Command Accepted signal was recently added to the CAMAC repertoire. The signal is transmitted in a manner similar to that of BQ. Each module generates an X response to every CAMAC Command addressed to that module. In essence, X='1' signals that the module received, and could and **did**, obey the command. An X='0' signals the converse. The Dataway X line collects signals from the modules of the crate. The Crate Controller Type A-1 wire-OR's the X signals onto the BX line of the branch highway. The BX signal is, of course, then available to the **branch driver.**

In many systems, an X='0' response will be an indication of a "serious" malfunction which should be brought to the attention of the computer at once. In such cases the computer should be interrupted upon BX=0.

In other systems, it is sufficient to set a BX flag, if BX=0, and let the software test the flag once in a while.

In a few cases, BX=0 is not necessarily an indication of malfunction. For example, older modules may not have X implemented. They always return X=0. Certain normal operations during Scan Address block transfers also return X=0. For example, in Scan Address, the Branch Driver may address an empty station in the process of discovering that it is empty.

To cope properly with these circumstances, the example of the branch driver in Fig. 3 permits a choice of three modes of BX monitoring. The modes are:

(1) Interrupt the computer immediately if a BX=0 is received during a command cycle;

(2) Save the BX response from the last command cycle in a flip-flop- and provide means for the computer program to test it--via "Skip on BX," for example;

(3) Completely ignore the BX response.

In the example, the computer program may select which of the three modes is in use by directing the proper instruction to the branch driver.

Final Note

The reader is reminded that this paper discusses some of the features that may be included in a CAMAC-computer interface. There are other possible features that have not been discussed. By the same token, not all the features that are discussed will necessarily be found in any particular design. Thus, the reader must not assume that all CAMAC systems behave like the hypothetical examples described. However, we hope that these descriptions will help him to recognize and use the features that his equipment possesses.

References

1. "CAMAC - A Modular Instrumentation System for Data Handling - Revised Description and Specification." Identical versions of this document have been issued by the U.S. AEC NIM Committee and by the European ESONE Committee. They are: U.S. AEC Report TID-25875, July 1972; and Euratom Report EUR-4100e, 1972.

2. "CAMAC - Organization of Multi-Crate System, Specification of the Branch Highway and CAMAC Crate Controller Type A. Identical versions of this document have been issued by the U.S. AEC NIM Committee, and by the European ESONE Committee. They are: U.S. AEC Report TID-25876, March 1972; and Euratom Report EUR-4600e, 1972.

3. F. A. Kirsten, "Operational Characteristics of the CAMAC Dataway." LBL-1575, February 1973 (See note)

4. F. A. Kirsten, "A Short Description of the CAMAC Branch Highway." LBL-1576, February 1973 (See note)

5. R. S. Larsen, "CAMAC Dataway and Branch Highway Signal Standards." (See note)

6. S. Dhawan, "The CAMAC Crate Controller Type A." (See note)

7. I. Flores, "Computer Organization." Prentice-Hall, Inc., Englewood Cliffs, N.J., 1969 Chapters 1,2,3,5 and 6.

8. J. J. Eichholz, F. R. Lenkszus, and M.G. Strauss, "Versatile CAMAC Crate Controller for Computer-Based Data Acquisition Systems." IEEE Trans. Nuclear Science vol NS-18, No. 1, Feb. 1971, pp.292-8.

9. S. Dhawan, "On the Design of CAMAC Branch Drivers." IEEE Trans Nuclear Science, vol NS-19, No. 1, Feb. 1972, pp.721-725.

10. H. Halling, K. Zwoll, and K. D. Muller, "Versatile PDP-11 CAMAC Crate Controller for Nuclear Data Acquisition and Processing." IEEE Trans. on Nuclear Science, vol NS-19, No. 1, Feb. 1972, pp.699-703.

11. J. A. Buchanan, H. V. Jones, "CAMAC Multi-Microprogrammed I/O Processor, IEEE Trans. on Nuclear Science, vol NS-19, No. 1, Feb. 1972, pp.682-688.

12. L. R. Biswell and R. E. Rojala, "Microprogrammed Branch Driver (MBD) for a PDP-11 Computer." Los Alamos Scientific Laboratory Report LA4916, April 1972 (Unpublished)

Note: References 3-6 will be published in the April 1973 issue of IEEE Transactions on Nuclear Science. This issue will be called "CAMAC Tutorial Issue."

TECHNICAL INFORMATION DIVISION
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720