A UNIVERSAL MICROPROCESSOR/ SERIAL CAMAC INTERFACE MODULE

P.J. Horne and O. Kaestner
INTRODUCTION
DESIGN PHILOSOPHY
DATA MEMORY
INTERFACE TO THE MICROPROCESSOR
4.1 Initiate command address
4.1.1 Single transfer mode
4.1.2 ZCI initiation
4.1.3 Load counter
4.1.4 Load memory address register
4.1.5 Load NAF command register
4.1.6 Block transfer mode
4.1.7 'A' scan transfer mode
4.1.8 Demand message initiation
4.2 Inhibit CAMAC (INC)
4.3 CAMAC response address (CRA)
4.4 Status word
CAMAC INTERFACE
5.1 CAMAC interface buffers/registers
5.2 Control of interface by CAMAC
5.2.1 CAMAC to memory data transfers (WRITE)
5.2.2 Memory to CAMAC data transfers (READ)
INTERFACE CONTROLLER
6.1 Microprogram sequencer
6.2 Microinstructions set
6.2.1 Field description
6.2.2 Instruction format
6.3 Instruction timing
DATA TRANSFER PROTOCOL AND TIMING DELAYS
7.1 The microprocessor expects data from the central computer
7.2 Central computer expects data from the microprocessor
7.3 System timing delays
REFERENCES
APPENDIX A: AN EXAMPLE OF CONTROLLING THE INTERFACE FROM A MICROPROCESSOR
APPENDIX B: MICROPROGRAMS
APPENDIX C: PARALLEL BRANCH HIGHWAY SYSTEM (STAR/CAMAC CRATE CONTROLLER AND TYPE A2 CONTROLLERS)
1. INTRODUCTION

The interface of microprocessor (μP) based auxiliary controllers to the CAMAC dataway\(^1\) is fast becoming the most expensive and time-consuming section of their design and implementation. This has arisen because of the large fall in price of μP's and their support chips (due to the possibility of large-scale integration with such circuits). Indeed, it is now possible to construct a microcomputer with 1K EPROM and 1K RAM with as few as eight integrated circuits (IC's), utilizing the INTEL 8080, and costing less than Sw.Fr. 500. On the other hand, the interface of such a microcomputer to CAMAC requires at least 45 small-scale (SSI) and medium-scale IC's to achieve a simple auxiliary controller capability. Furthermore, if one requires data transfers, in a block mode, between the microcomputer and the computer controlling the CAMAC system (control computer) one needs a direct memory access (DMA) control unit as part of the interface. This significantly increases the complexity of the interface and fixes it to one particular μP. One can, of course, have an external memory module (also in the CAMAC crate) which may be accessed by both computers, thereby removing the need for a DMA controller but requiring another CAMAC station.

With no guidelines for interfacing μP's to CAMAC available, duplication of effort is unavoidable and, probably worse, the multiplicity of designs makes maintenance in a large system extremely difficult. It is for this reason and that described in the previous paragraph that an interface capable of connecting to many μP's (be they 8- or 16-bit) is required.

This report describes the specifications and implementation of an interface module capable of connecting a wide variety of μP's to the serial CAMAC crate system\(^1\).

2. DESIGN PHILOSOPHY

The advanced design philosophy of microcomputer systems, where high-speed, fixed hardware, is replaced by multipurpose IC's whose function may be altered by programming, has yet to reach CAMAC interface design. In such interfaces the design must be realized by wiring together many SSI and MSI integrated circuits.
There are a few programmable interface chips available, such as the Programmable Peripheral Interface (PPI) of INTEL and the Peripheral Interface Adaptor (PIA) of Motorola: however, they do not readily adapt to the CAMAC dataway and require much programming to perform simple data transfers.

To circumvent the hardware complexity of the interface circuitry one can follow the basic philosophy of μP's, the replacement of hardware by software. This calls for the introduction of a programmable controller, for example, a microprogrammable sequencer. The microinstructions could directly exercise control in a well-structured design.

Some of the relatively slow instructions (∼2 μsec cycle time) required by the μP to access the CAMAC dataway may now be replaced by a sequence of fast (200 nsec cycle time) microinstructions. Thus a simplification of the μP's software interface is an additional benefit.

As one finds that the majority of data transfers in CAMAC only use 16 bits, due to the predominance of the 16-bit minicomputer, the interface will be laid out only for this word size.

A block diagram of the interface module is shown in Fig. 1, where the main functional blocks are:

i) 1K × 16 bit random access memory (RAM).

ii) Interface to the μP.

iii) CAMAC interface buffers.

iv) Interface controller.

The RAM acts as a buffer memory between CAMAC and the μP such that all data transfers between them must temporarily be stored here. To the controller and CAMAC the memory is configured as 1K × 16 bits, but to the μP it may also be accessed as 2K × 8 bits. An important use of this memory is for the storing of CAMAC commands to be initiated by the μP and executed by the interface controller.

The interface to the μP contains control and status registers necessary for the μP to initiate the stored commands and read the respective responses. An interrupt signal is provided to enable the controller to inform the μP when a previously initiated command has been terminated.
CAMAC interface buffers are, as their name implies, required to buffer data between the internal data and control bus and the dataway. Operation of these buffers is the responsibility of the controller, as is the generation of the two dataway timing signals S1 and S2.

Finally, the heart of the interface module is the controller which consists of a microprogrammed sequencer and its associated microprogram memory. This controller performs such tasks as initiating dataway cycles on command from the µP, giving the required responses to CAMAC commands issued by the central computer and interrupting the µP on the occurrence of a LAM.

By replacing interface hardware by software, in the form of the microprogrammed sequencer, one is able to reduce the number of IC's, readily structuring its design, and providing a more versatile module.

A more detailed description of the interface module will now be given.

3. DATA MEMORY (1K x 16 BIT RAM).

The 1K words of memory may be accessed by either the µP or the controller, both having control of the address and data lines. To enable both 16-bit and 8-bit µP to access this memory a different configuration will be required. This is achieved by using a PROM (Signetics N825123, 50 nsec PROM) to enable the memory as 1K words or 2K bytes (see Fig. 2) and a set of dual in-line switches to compress the 16-bit data bus to 8 bits.

When the µP is not accessing this memory the tri-state buffers (see Fig. 2) are in the high impedance state, thus isolating it from the memory. The controller may now access the memory without its address and data lines conflicting with those of the µP. It is, however, possible that the controller may want to access the memory at the same time as the µP; in this case it is the µP that has priority. This priority arbitration isinstigated by means of the memory enable (µPMEMEN) signal, generated whenever the µP addresses the interface memory.

To prevent the µP from interrupting controller-memory operations started by a CAMAC command from the central computer, a certain protocol should be followed (see Section 7).
The top 32 words of this memory are reserved for the μP to store (i) CAMAC commands and the associated data, (ii) an address for the memory address register (MAR) (see Section 4.1.2), (iii) data for the counter (CNTR) (see Section 4.1) and (iv) a demand message to the serial crate controller. One word of this area is also reserved for the controller to store a LAM pattern (see Section 4.3).

A suggested start address for this interface memory (see Fig. 3) as seen by the μP, is F00016 (= 60K), as this is unlikely to interfere with a program running on the microcomputer. This address can, however, be altered by the user, for the μP's which do not have a 64K addressing range (such as the TMS 9900).

4. INTERFACE TO THE MICROPROCESSOR

In order that as many μP's as possible are able to utilize this interface, only address, data and memory read and write lines should be used for its control. With only these lines used, the interface module will 'look' like a block of memory to the μP: therefore all control is effected by accessing predefined memory locations. There are four addresses reserved for control of the interface; these are

i) Initiate command address (ICA) (address = F80016)

ii) Inhibit CAMAC (INC) (address = F80216)

iii) CAMAC response address (CRA) (address = F80416)

iv) Status word (STATUS) (address = F80616).

A diagram of this interface is shown in Fig. 4.

4.1 Initiate command address

This address is accessed (written to) when the μP wants to initiate a microprogram stored in the controller's microprogram memory. These microprograms are concerned with generating CAMAC commands, whose variables are stored in the interface memory, and data transfers on the interface.

To initiate a particular microprogram, data equal to its start address must be written to the ICA (see Fig. 4). The data corresponding to the possible interface operations are:

i) Data = 0 through 8 A CAMAC dataway cycle will be initiated in single transfer mode (see Section 4.1.1).
ii) Data = 9  
Generation of the dataway Common Control Signals Z, C and I.

iii) Data = 10  
The counter is loaded (see Section 4.1.2).

iv) Data = 11  
The MAR is loaded (see Section 4.1.3).

v) Data = 12  
The 'NAF' CAMAC COMMAND REGISTER is loaded (see Section 4.1.4).

vi) Data = 13  
A CAMAC dataway cycle is initiated in block transfer mode (see Section 4.1.5).

vii) Data = 14  
A CAMAC dataway cycle is initiated in 'A Scan Mode'(Section 4.1.6).

viii) Data = 15  
The SGLE register is loaded and a demand message initiated (see Section 4.1.7).

Prior to initiating one of the above operations one should have stored the requisite data in the correct memory location. The memory locations corresponding to the microprogram called are given in Table 1.

**Table 1**

Microprogram parameter addresses

<table>
<thead>
<tr>
<th>Microprogram</th>
<th>ICA(i)</th>
<th>Data store addresses*</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>i = data</td>
<td>(DSA(i))</td>
<td></td>
</tr>
<tr>
<td>Single transfer mode</td>
<td>0</td>
<td>F7C0, F7C2</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>F7C4, F7C6</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>F7C8, F7CA</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>F7CC, F7CE</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>F7D0, F7D2</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>F7D4, F7D6</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>F7D8, F7DA</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>F7DC, F7DE</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>F7E0, F7E2</td>
<td>NAF, DATA</td>
</tr>
<tr>
<td>ZCI</td>
<td>9</td>
<td>F7E4, F7</td>
<td>ZCI</td>
</tr>
<tr>
<td>Load counter</td>
<td>10</td>
<td>F7E8</td>
<td>Count</td>
</tr>
<tr>
<td>Load MAR</td>
<td>11</td>
<td>F7EA</td>
<td>Address</td>
</tr>
<tr>
<td>Load NAF</td>
<td>12</td>
<td>F7EC</td>
<td>N, A, F</td>
</tr>
<tr>
<td>Block transfer</td>
<td>13</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>A scan</td>
<td>14</td>
<td>F7F0</td>
<td>DATA</td>
</tr>
<tr>
<td>Initiate demand message</td>
<td>15</td>
<td>F7F2</td>
<td>Demand message</td>
</tr>
</tbody>
</table>

*) All address are in Hexadecimal.
4.1.1 Single transfer mode

In this mode of operation a CAMAC command (N,A,F) is executed and its corresponding data transferred between the dataway and the interface memory. For the μP to initiate this dataway cycle, the following sequence of operations must be adhered to:

i) The μP writes the required N,A,F as a 16-bit word (or two 8-bit bytes) into one of the 9 specified DSA's (see Table 1). The bit definition of this word is:

<table>
<thead>
<tr>
<th>BYTE 2 (MSB)</th>
<th>BYTE 1 (LSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N16, N8, N4, N2, N1, F16, F8, F4, F2, F1, A8, A4, A2, A1</td>
<td></td>
</tr>
<tr>
<td>15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0</td>
<td></td>
</tr>
</tbody>
</table>

ii) If the command is a CAMAC write the 16-bit data word should be written to the next sequential (DSA(N) + 2, \(0 \leq N \leq 8\) word (or pair of bytes).

iii) The command is now initiated by writing to the ICA with data corresponding to the DSA used for storing the NAF. (ICA(\(\phi\)) - ICA(8)).

On termination of the dataway cycle the interface controller will interrupt the microprocessor. If all was well only the EOC bit will be set in the status word; otherwise an error bit will be set (see Section 4.4). In this mode of operation the μP must test for X and Q by reading the CAMAC response address (CRA) (section 4.3).

4.1.2 ZCI initiation

Generation of the dataway common control signals (Ref. 1, pp. 71-72)

Initialize (Z), Clear (C) and Inhibit (I) is controlled by three bits in DSA(9), bits 0, 1, and 2, respectively. With the required bits set in DSA(9) a dataway cycle will be initiated on the μP writing to ICA(9) with data = 9.

4.1.3 Load counter

This counter forms part of the microprogrammed interface controller and is utilized by it for loop control during block transfers. To initiate a load of this register the μP should:

i) Write the count value (\(\phi - FF_{16}\)) into DSA(10).

ii) Initiate the operation by writing Data = 10 to the initiate command address (ICA(10)).
4.1.4 Load memory address register (MAR)

The memory address register and its associated buffer (MARB) are used by the interface controller for latching an address during memory read/write operations. A latched MARB is necessary because one needs to load the MAR from memory during block transfer and 'A' scan mode microprograms.

To load the MAR the following two operations should be performed by the μP.

i) Write the address (φ - 3FF₁₆) into DSA(11).

ii) Initiate the operation by writing to ICA with data = 11.(ICA(11))

The start address of the interface memory as seen by the controller is 0000₁₆ and corresponds to the μP address F000₁₆.

4.1.5 Load NAF command register

This register interfaces to the CAMAC dataway and holds the N, A and F data for the selected CAMAC command. Its loading is similar to that for the counter and MAR, as the μP first stores the NAF in DSA(13)) (see Section 4.1.1 for bit specification) and initiates the operation by writing to ICA with data = 12. (ICA(12))

4.1.6 Block transfer mode

One utilizes this mode of operation to transfer blocks of data from the interface memory to a CAMAC module or vice versa. Certain error conditions may occur during the execution of a data transfer and should be tested for by the interface controller, which should then inform the μP. These error conditions are:

i) No X response from the module.

ii) No Q response from the module.

When either error occurs the interface controller will stop the block transfer, load the remainder of the count value into the memory at DSA(14), and interrupt the μP with the relevant error bit set in the status word (see Section 4.4).

The following steps should be taken by the μP for block transfers.

i) Load the NAF command into the NAF register (see Section 4.1.4).

ii) Load MAR to point to the first word of the block's memory buffer (see Section 4.1.4).
iii) Load the number of words in the block to be transferred into the counter (see Section 4.1.2).

iv) Initiate block transfer command by writing to ICA with data = 13 (ICA(13)).

If the µP is interrupted because of an error condition it should read the status word and the remaining count at DSA(13). It is now up to the µP to decide what action to take. If the block transfer was successful then the µP will again be interrupted, but this time only the end of cycle (EOC) bit will be set in the status word (see Section 4.4).

4.1.7 'A' scan transfer mode

In this mode of operation the µP is able to scan the 16 possible 'A' addresses in a CAMAC module. The scanning of these addresses is performed by the interface controller, which increments the 'A' field of the NAF command after each successful dataway cycle. The sequence of events that the µP must follow are:

i) Load NAF command register (see Section 4.1.4).

ii) Load the data (for a CAMAC write) into DSA(14).

iii) Initiate the command by writing to ICA with data = 14. (ICA(14)).

If no Q response is obtained from this CAMAC command the address (A) lines will not be incremented, thus enabling the µP to repeat the command if necessary.

4.1.8 Demand message initiation

When the µP needs to communicate with the central computer it does so by means of initiating a demand message via the serial crate controller.

Within the crate the µP, and its interface (the ensemble is normally referred to as an auxiliary crate controller), has the responsibility of handling LAM's. Whilst most of them can be serviced locally by the auxiliary controller (thus reducing interrupt servicing overheads on the control computer) there will be a few that are beyond its capability. The occurrence of one of these LAM's should therefore be signalled to the central computer by means of a demand message. The message may simply be a number (5 bits) representing the LAM which occurred.

To initiate a demand message the µP should load DSA(15) with the required 5-bit demand message (bit 0-4), known as the SCLE field\(^2\) and the 5th bit set to 1 to enable the demand message Initiate (DMI) bit. The ICA should then be
written to, with data = 15. On this action the interface controller will load
the SGLE register and initiate the demand message. However, if a previous message
is still resident in the serial crate controller (i.e. it has not yet been trans-
mitted to the central computer) the new demand message will not be initiated. In
this case the interface controller will interrupt the μP and set the demand busy
(DBSY) bit in the status word. It is now the responsibility of the μP to wait and
attempt to transmit a demand message at some later time or transmit a hung demand
(Ref. 1, p 81). The hung demand informs the central computer of the presence of
an unserviced message and requires that the bits of the SGLE field be set to logic '1'.

4.2 Inhibit CAMAC (INC)

The inhibit CAMAC bit is set by the μP when it requires that CAMAC should
not access the interface. This is done by writing to address F80216 with data = '1'
to set the inhibit bit and 0 to reset it.

The need for such a control bit is to allow the μP uninterrupted access to the
interface memory when it is performing data manipulation, etc. During this time,
should the central computer initiate a CAMAC command to the interface module it
will only receive an X response. When the μP has some valid results for the cen-
tral computer it can reset the Inhibit CAMAC bit and wait until the data has been
acquired before proceeding.

4.3 CAMAC response address (CRA)

As its name implies this address (F80416) is used for holding the X and Q
response to the last dataway cycle initiated by the auxiliary controller (interface
module and μP). By reading from this address the μP may test the state of the X
and Q bits (bit 0 and 1, respectively).

4.4 Status word

The status word is located at address F80616 and is utilized by the interface
controller to inform the μP of:

i) a LAM demand in the crate;

ii) a successful access to the interface module from CAMAC;

iii) the result of the last initiated microprogram.

The interface controller interrupts the μP when there is a change in the contents
of this status word. To acquire the status information the μP reads the contents of address F80616, which are:

- **Bit 0**: Data available (DTV), set on the central computer writing to the interface memory.
- **Bit 1**: Data read (DTR), set on the central computer reading from the interface memory.
- **Bit 2**: LAM, set on the occurrence of a LAM demand in the crate.
- **Bit 3**: End of cycle (EOC), set at the successful termination of an auxiliary controller CAMAC command.
- **Bit 4**: No Q response (NQ) during a CAMAC command.
- **Bit 5**: No X response (NX) during a CAMAC command.
- **Bit 6**: Demand busy (DBSY) is set when the interface controller finds the bit of the same name set in the serial crate controller, during demand message initiation.
- **Bit 7**: Auxiliary lockout (AUX LOC) is set when the interface controller finds this bit set in the serial crate controller.

5. **CAMAC INTERFACE**

The CAMAC interface includes the necessary hardware for the interface controller to access the dataway (Fig. 5), and the hardware and firmware (microprograms) for CAMAC to control the interface.

5.1 **CAMAC interface buffers/registers**

Figure 5 shows the necessary buffers/registers that the interface controller requires for performing CAMAC commands and responding to CAMAC commands from the central computer. To avoid confusion between which source (central computer or interface controller) generated a CAMAC command, those generated by the interface/μP ensemble will be known as auxiliary CAMAC commands.

The function of the buffers/registers are:

- **Receive buffer**: This buffer is used during auxiliary CAMAC read operations and is enabled by the control line REC.
Read buffer/register

The read buffer/register is used as a temporary store for read data during a CAMAC read command. This buffer/register is necessary because the interface controller cannot access memory and have the data ready before CAMAC S1 timing (see, Section 6). The control lines are RED to enable the data onto the dataway and LRED to load the register.

Write/Transmit buffer

This bi-directional buffer enables 16-bit data onto the Write lines (W1-16) during an auxiliary write command (Transmit) and onto the data bus, from the write lines, during a CAMAC write command. The control lines are WRT for a CAMAC write enable and TRM for an auxiliary CAMAC write enable.

NAF command buffer/register

This buffer/register consists of three sections, which are used during all auxiliary CAMAC commands. (i) The N buffer/register which controls the N(1-25) lines, via the SGLE connector, for accessing CAMAC stations; (ii) the F buffer/register for controlling the F1-16 lines; and (iii) the 'A' buffer/register which controls the A1-8 address lines. The 'A' buffer/register also has the possibility of being incremented on command from the interface controller. The control lines are NAF for enabling the data onto the dataway LNAF for loading the buffer/register and LNCNAF for incrementing the address lines. The two signals AFl6(F16) and AF8(F8) are used to inform the interface controller of an auxiliary Read or Test command.
LAM buffer
The LAM buffer patches 16 possible LAM demands from the SGLE connector onto the data bus and is enabled by means of the RLAM line.

SGLE field buffer/register
This register holds the demand message (the 5 SGLE bits) for the SGLE connector, it is loaded by a 0 to 1 transition of the LSGLE signal.

5.2 Control of interface by CAMAC
To the central computer the interface module will be transparent, that is only the 1K words of memory will be seen. A data transfer to or from this memory requires a minimum sequence of three CAMAC commands (see Table 2).

5.2.1 CAMAC to memory data transfers (WRITE)
To perform a CAMAC write to the interface memory from the central computer one should adhere to the following sequence of commands

i) F(16)A(1)
Load the MAR with address (0-1K) of the memory location where data is to be stored. The write data W1-W10 corresponds to address bits A0-A9.

ii) F(16)A(0)
Write the required data into memory at the address pointed to by the contents of MAR.

iii) F(24)A(0)
Inform the μP of the termination of a write sequence to the interface memory.

After each F(16)A(0) command the contents of the MAR are incremented by the interface controller enabling the central computer to write a block of data to the memory.

5.2.2 Memory to CAMAC data transfers (READ)
When the central computer wishes to read from the interface memory the following sequence of commands should be utilized.

i) F(16)A(12)
Load the MAR with the address of the memory location from which data is to be read. This command also starts a micro-program in the interface controller to load the required data into the RREG ready for the next read command.
ii) F(0)A(0)

Read the contents of the RREG, increment the MAR and start the microprogram to read the next word of memory (pointed at by MAR) into the RREG.

iii) F(24)A(1)

Inform the µP of the termination of a read sequence from the interface memory.

Once again to read a block of data from the interface memory only the F(0)A(0) command need be repeated.

Table 2

Decoded CAMAC commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F(0)A(0)</td>
<td>X and Q if Inhibit CAMAC bit is zero</td>
<td>Read RREG</td>
</tr>
<tr>
<td>F(16)A(0)</td>
<td>X and Q if Inhibit CAMAC bit is zero</td>
<td>Write to memory 8(MAR)</td>
</tr>
<tr>
<td>F(16)A(1)</td>
<td>X and Q if Inhibit CAMAC bit is zero</td>
<td>Write to MAR, when next command will be F(16)A(0)</td>
</tr>
<tr>
<td>F(16)A(2)</td>
<td>X and Q if Inhibit CAMAC bit is zero</td>
<td>Write to MAR, when next command will be F(0)A(0)</td>
</tr>
<tr>
<td>F(24)A(0)</td>
<td>X and Q if Inhibit CAMAC bit is zero</td>
<td>Inform µP of the termination of a write to memory sequence</td>
</tr>
<tr>
<td>F(24)A(1)</td>
<td>X and Q if Inhibit CAMAC bit is zero</td>
<td>Inform µP of the termination of a read from memory sequence</td>
</tr>
</tbody>
</table>

6. INTERFACE CONTROLLER

The heart of this interface is an Am2909 microprogram sequencer and its associated 256 x 24 bit PROM (see Fig. 6) which form the centre of the microprogrammed controller. The function of this controller is the generation of the necessary control signals for (i) correct response to the decoded CAMAC commands and (ii) performing those commands requested by the µP. One further responsibility of the controller is the checking for the occurrence of a LAM demand in the crate and informing the µP when one occurs.
The interface controller manipulates the three other main interface blocks (memory, CAMAC buffers and μP interface) by means of a control bus. Data on this control bus at any particular instance is a function of the microinstruction word, held in its respective register. Due to this direct relation between control bits and the particular microinstruction being executed the term 'microprogrammed controller' is valid rather than 'programmed controller'.

6.1 Microprogram sequencer

The Am2909 microprogram sequencer was chosen for this application due to its simplicity of operation and its speed (16.6 MHz max. clock rate). This bipolar sequencer is a cascadable 4-bit slice device, i.e. one requires two such sequencers in parallel to address 256 words of memory. A complete description of this device may be found elsewhere; however, a brief description of the functions used by this controller application will now be given.

The sequencer contains a four-input multiplexer for selection of the next microinstruction address source; these are (i) an address register, (ii) direct inputs, (iii) a microprogram counter, and (iv) a four word Push/POP register file. Selection of one of these four sources is obtained by setting the relevant code (Table 3) on the $S_0$ and $S_1$ lines (see Fig. 5). In this application the direct inputs (D bus) are connected to either four CAMAC command lines (A1, A2, F8, F16) or the four ICA data lines (see Figs. 4 and 6). Selection between these two sources is made by means of the CAMAC N line and the START MICROPROGRAM line.

The address register (R bus) is connected to an address field in the microinstruction word used for the destination address in direct jumps.

The four-word register file allows subroutine calls, nested up to four levels, and temporary data storage. Use of this file is controlled by the File Enable (FE) and Push/Pop (PUP) lines. In this application only one level of subroutine nesting is envisaged.

The microprogram counter holds the next sequential address, which in most cases will be the incremented present address (that on the I bus, see Fig. 6). The other sources of the next (microinstruction) address are selected during (i) a JUMP to an address on the R bus, (ii) a subroutine call or return, and (iii) a 'forced' call to the address on the D bus by CAMAC or the μP.
Over-all control of the microprogram sequencer is exercised by the four following bits: $S_0$, $S_1$, FE, and PUP. They are included in the microinstructions as bits 20 to 23 (see Section 6.2) and their possible combinations in this application are given in Table 4.

### Table 3
Microprogram sequencer control lines

<table>
<thead>
<tr>
<th>$S_0$ $S_1$ FE PUP</th>
<th>Mnemonic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0</td>
<td>NOP</td>
<td>Microprogram counter is used as next address</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>JSR</td>
<td>PUSH microprogram counter and JUMP to address or R bus</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>JMP</td>
<td>JUMP to address on R bus</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>RTS</td>
<td>JUMP to address on stack and POP stack</td>
</tr>
</tbody>
</table>

#### 6.2 Microinstruction set

A microinstruction set has been developed for the interface controller which eases the task of writing the microprogram. Each microinstruction word is 24 bits long and is divided into a number of control fields, the selection of these fields being performed by an operational code (bits (16,19)). A block diagram showing the control lines generated by each field is given in Fig. 7 and the function of each control line in Table 4.

There are ten possible microinstructions for this application which are:

i) **Move a word (MOV)**

OP. CODE: 210000

```
23 22 21 20 19 18 17 16 15 14 13 12 10 9 8 7 6 5 4 3 2 1 0
0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 S D 0 0 0
```

This instruction moves the contents of the source to the destination.

ii) **Branch if condition is true (BRIF)**

OP. CODE: 220000

```
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 1 0 0 0 1 0 | R | 0 0 0 0 0 T
```
Test the condition ('on' or 'off') of the bit specified by the T field and branch to the address in the R field if it is true (on).

iii) Load destination with immediate data (LDI)

**OP. CODE:** 230000

```
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 0 0 1 0 0 0 1 1 DA 1 D 0 0 0
```

Load the destination with the data specified in the DA field.

iv) Increment or decrement (INCR/DECR)

**OP. CODE:** 240000

```
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 D 0 0 0
```

This instruction, depending on the destination specified, will increment or decrement the contents of the destination.

For example, one may INCR MAR but not INCR CNTR (it can only be decremented).

v) Interrupt the microprocessor (INT)

**OP. CODE:** 250000

```
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 0 0 1 0 0 0 1 0 1 ST 0 0 0 0 0 0 0 0
```

The microprocessor will be interrupted and the status register loaded with the bit pattern specified in the ST field.

vi) CAMAC operation (CAM)

**OP. CODE:** 280000

```
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 CA S1 S2 0 0 0
```

Perform the CAMAC operation specified in the CA field and enable the S1 and S2 bits when set. This instruction may be merged with the BRIF instruction for certain CAMAC commands, e.g., CAM MEM(R) and BRIF AUXLOC, ADDRESS (CODE: 2A0000).

vii) No operation (NOP)

**OP. CODE:** 200000

```
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```
viii) Jump to subroutine (JSR)

OP. CODE: 900000

Jump to the subroutine who's address is specified in the R field and push the contents of the microprogram counter.

ix) Jump unconditional (JMP)

OP. CODE: A00000

Jump to the address specified in the R field.

x) Return from subroutine (RTS)

OP. CODE: 400000

Jump to the address on the stack and pop stack.

6.2.1 Field descriptions

i) Source (Si)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Mnemonic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6</td>
<td>MEM</td>
<td>The contents of the memory location whose address is held in MAR</td>
</tr>
<tr>
<td></td>
<td>LAM</td>
<td>The contents of the LAM register</td>
</tr>
<tr>
<td></td>
<td>CNTR</td>
<td>The contents of the counter</td>
</tr>
<tr>
<td></td>
<td>MAR</td>
<td>The contents of the memory address register</td>
</tr>
</tbody>
</table>
### ii) Destination (D_i)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Mnemonic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>MEM</td>
<td>The memory location whose address is held in MAR</td>
</tr>
<tr>
<td>0 0 1</td>
<td>MAR</td>
<td>The memory address register</td>
</tr>
<tr>
<td>0 1 0</td>
<td>NAF</td>
<td>The NAF command register</td>
</tr>
<tr>
<td>0 1 1</td>
<td>SGLE</td>
<td>The SGLE register</td>
</tr>
<tr>
<td>1 0 0</td>
<td>CNTR</td>
<td>The counter</td>
</tr>
<tr>
<td>1 0 1</td>
<td>RREG</td>
<td>The CAMAC read register</td>
</tr>
<tr>
<td>1 1 0</td>
<td>MARB</td>
<td>The memory address register buffer</td>
</tr>
</tbody>
</table>

### iii) Test (T_i)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Mnemonic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>AUXLOC</td>
<td>Test the state of the auxiliary lockout bit</td>
</tr>
<tr>
<td>0 0 1</td>
<td>AUXREAD</td>
<td>Test the state of the auxiliary read bit</td>
</tr>
<tr>
<td>0 1 0</td>
<td>AUXTEST</td>
<td>Test the state of the auxiliary test bit</td>
</tr>
<tr>
<td>0 1 1</td>
<td>CNTRZ</td>
<td>Test the state of the counter = zero bit</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Q</td>
<td>Test the state of the Q bit</td>
</tr>
<tr>
<td>1 0 1</td>
<td>DBSY</td>
<td>Test the state of the demand BUSY bit</td>
</tr>
<tr>
<td>1 1 0</td>
<td>X</td>
<td>Test the state of the X bit</td>
</tr>
<tr>
<td>1 1 1</td>
<td>LAM</td>
<td>Test the state of the LAM bit</td>
</tr>
</tbody>
</table>
iv) CAMAC (CAₖ)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Mnemonic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 5 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>MEM(R)</td>
<td>Auxiliary receive, read from the dataway to memory pointed at by contents of MAR</td>
</tr>
<tr>
<td>0 0 1</td>
<td>MEM(W)</td>
<td>Write to memory location whose address is held in MAR</td>
</tr>
<tr>
<td>0 1 0</td>
<td>MAR</td>
<td>Memory address register (write only from CAMAC)</td>
</tr>
<tr>
<td>0 1 1</td>
<td>RREG</td>
<td>Read the contents of the CAMAC read register</td>
</tr>
<tr>
<td>1 0 0</td>
<td>MEM(T)</td>
<td>Read the contents of memory location pointed at by MAR and transmit to CAMAC (auxiliary transmit)</td>
</tr>
<tr>
<td>1 0 1</td>
<td>NAF</td>
<td>Enable the NAF command register/buffer onto the dataway</td>
</tr>
<tr>
<td>1 1 0</td>
<td>ZCI</td>
<td>Generate the dataway common control signals</td>
</tr>
</tbody>
</table>

v) Status (STᵢ)

<table>
<thead>
<tr>
<th>15 14 13 12 11 10 9 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST₁ ST₂ ST₃ ST₄ ST₅ ST₆ ST₇ ST₈</td>
</tr>
</tbody>
</table>

- Data available (DTV) |
- Data read (DTR) |
- LAM demand (LAM) |
- End of cycle (EOC) |
- No Q response (NQ) |
- No X response (NX) |
- Demand busy (DBSY) |
- Auxiliary lockout (AUXLOC) |

6.2.2 Instruction format

- MOV Sᵢ, Dᵢ
- BRIF Tᵢ, Rᵢ
- LDI DAᵢ, Dᵢ
- INCR Dᵢ
- DECR Dᵢ
- INT STᵢ
- CAM CAᵢ, S2, S1 (CAM CAᵢ, S2, S1 & BRIF Tᵢ, Rᵢ)
- JSR Rᵢ
- JMP Rᵢ
- RTS
- NOP
### Table 4
#### Control lines

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>WADD</td>
<td>Enable immediate (from microinstruction) address field onto data bus</td>
</tr>
<tr>
<td>INCMAR</td>
<td>Increment the contents of MAR</td>
</tr>
<tr>
<td>LMAR</td>
<td>Load MAR from the data bus</td>
</tr>
<tr>
<td>LMARB</td>
<td>Load MARB from MAR</td>
</tr>
<tr>
<td>MEMEN</td>
<td>Enable memory address onto address lines</td>
</tr>
<tr>
<td>CMEMR</td>
<td>Read from the interface memory</td>
</tr>
<tr>
<td>CMEMW</td>
<td>Write to the interface memory</td>
</tr>
<tr>
<td>INT</td>
<td>Interrupt (\mu)P and load status word</td>
</tr>
<tr>
<td>REC</td>
<td>Enable R1-R16 onto data bus</td>
</tr>
<tr>
<td>RED</td>
<td>Enable read register (RREG) contents onto R1-R16 lines</td>
</tr>
<tr>
<td>LRED</td>
<td>Load RREG from data bus</td>
</tr>
<tr>
<td>WRT</td>
<td>Enable W1-W16 onto data bus</td>
</tr>
<tr>
<td>TRM</td>
<td>Enable data bus onto W1-W16</td>
</tr>
<tr>
<td>NAF</td>
<td>Enable contents of NAF buffer/register onto dataway</td>
</tr>
<tr>
<td>INCNAF</td>
<td>Increment the 'A' field of NAF buffer/register</td>
</tr>
<tr>
<td>LNAF</td>
<td>Load NAF buffer/register from data bus</td>
</tr>
<tr>
<td>RLAM</td>
<td>Read LAM buffer onto data bus</td>
</tr>
<tr>
<td>S1</td>
<td>Initiate CAMAC S1 timing pulse</td>
</tr>
<tr>
<td>S2</td>
<td>Initiate CAMAC S2 timing pulse (after a delay of 100 nsec)</td>
</tr>
<tr>
<td>LCNTR</td>
<td>Load counter from data bus</td>
</tr>
<tr>
<td>RCNTR</td>
<td>Read counter onto data bus</td>
</tr>
<tr>
<td>DECCNTR</td>
<td>Decrement the contents of the counter</td>
</tr>
</tbody>
</table>
6.3 Instruction timing

There is one, two-phase, 5 MHz clock which provides all timing for the controller. Phase 1 ($\phi_1$) drives the microprogram sequencer and phase 2 ($\phi_2$) the micro-instruction register. Since the access time of the PROM (N825114 256 × 8) micro-program memory is 60 nsec and the delay time from clock ($\phi_1$) to output of the Am2909 is ~ 40 nsec, the microinstruction can be strobed into its register only after this composite (~ 100 nsec) delay. The $\phi_2$ clock is therefore delayed by 100 nsec from the $\phi_1$ clock catering for the worst case delays of the sequencer and PROM.

This instruction cycle time is easily related to the timing of the CAMAC dataway, so that it is simple for the interface controller to generate CAMAC timing signals, such as S1 and S2. However, as S2 must be generated 700 nsec after the start of a CAMAC dataway and with a 200 nsec cycle time, this is not possible; a 100 nsec delay must be provided, i.e. the S2 signal is generated at the 600 nsec timing, after the start of a CAMAC dataway cycle, by the interface controller and a hardware delay of 100 nsec prevents it from being generated until 100 nsec later.

To enable the use of the Am9130 (1K × 4 bit 200 nsec) RAM memory, all memory reference (read) microinstruction must be repeated twice. This is due to the $\phi_1$ clock being used for strobing data into the required destination register during such microinstructions, for example, MOV, MEM, NAF, and the access time of the memory being 200 nsec. By prolonging this instruction (i.e. repeating it) data will be valid for the next $\phi_1$ strobe.

7. DATA TRANSFER PROTOCOL AND TIMING DELAYS

To prevent ambiguous operations of this interface when transferring data between the central computer and the µP and vice versa a certain protocol must be followed.

7.1 The µP expects data from the central computer

In this case the µP should wait until it is informed by the interface controller, via the DTV status bit, that a data transfer to the interface memory has been terminated. If the data is located in a contiguous block of memory, its start address must either
ii) be known \textit{a priori} to the \textit{\mu}P, or

iii) be pointed at by MAR (having been loaded by the last CAMAC compound), or

The second and third cases would be utilized when the data in the memory is to be written immediately to another module in the CAMAC crate. It is, however, envisaged that the first case will be the most generally used one, with the data block's size and start address being specified prior to implementation of the particular application.

During the time that the \textit{\mu}P is utilizing the interface the central computer must be prevented from interrupting it, this is achieved by the \textit{\mu}P setting the INHIBIT CAMAC bit. With this bit set the interface module will only give X response to all CAMAC commands signalling that they were not executed. Thus if the central computer wishes to load a second block of data into the memory, the initial command (see Section 5.2.1) should be repeated until a Q response is received.

7.2 Central computer expects data from the \textit{\mu}P

In this case the \textit{\mu}P writes the data into the interface memory and informs the central computer of its presence by either transmitting a demand message or 'dropping' the Inhibit CAMAC bit. The transmitting of a demand message is the more efficient solution as it saves the central computer from repeated testing of the state of the interface module (i.e. is the INHIBIT CAMAC bit set).

The start address of the block of data should either be known \textit{a priori} to the central computer or be pointed at by the MAR. If it is always held in the MAR the central computer need not load this register (see Section 5.2.2) prior to reading the data. It must, however, inform the \textit{\mu}P that the data has been read by sending the CAMAC command F(24)A(1) (see Section 5.2.2).

7.3 System timing delays

In a serial crate controller there is the possibility of preventing auxiliary controllers (interface/\textit{\mu}P ensemble) access to the dataway by setting an 'auxiliary lockout' bit (AUXLOC). This bit is set automatically on the crate controller recognizing its address in an incoming message (Ref. 1, p. 81) and remains 'on' until the required dataway cycle has terminated or an error in the message detected.
The interface controller tests the condition of the AUXLOC bit before each µP requested dataway cycle, and if set will not start the cycle. The µP is informed of this situation by the interface controller reading an interrupt and setting the AUXLOC (see Section 4.4) bit in the status word.

In a 5 MHz byte serial mode crate system the maximum time that the AUXLOC bit will be set is 4.7 µsec; thus the µP should wait for this delay before re-initiating the CAMAC command. In systems which have slower clock rates or use the bit serial mode the delay will be proportionately longer, making it necessary for the µP to spend more time waiting for the execution of CAMAC commands. However, with an intelligent auxiliary controller in the system the number of access by the central computer to the crate should be few, keeping to a minimum the number of times the µP is delayed.

* * *

REFERENCES


Fig. 1 Block diagram of interface
Fig. 2 Interface memory
Interface controller word address

\[ \begin{array}{|c|c|c|}
\hline
00016 & F000_{16} \\
\hline
\end{array} \]

\textbf{DATA TRANSFER MEMORY}
\textbf{FOR BLOCK AND 'A' SCAN MODES}

\begin{tabular}{|c|c|c|}
\hline
3E0 & DSA(0) 4 bytes & F7C0 \\
3E2 & DSA(1) 4 bytes & F7C4 \\
3E4 & DSA(2) 4 bytes & F7C8 \\
3E6 & DSA(3) 4 bytes & F7CC \\
3E8 & DSA(4) 4 bytes & F7D0 \\
3EA & DSA(5) 4 bytes & F7D4 \\
3EC & DSA(6) 4 bytes & F7D8 \\
3EE & DSA(7) 4 bytes & F7DC \\
3F0 & DSA(8) 4 bytes & F7E0 \\
3F2 & DSA(9) 4 bytes & F7E4 \\
3F4 & DSA(10) 2 bytes & F7E8 \\
3F5 & DSA(11) 2 bytes & F7EA \\
3F6 & DSA(12) 2 bytes & F7EC \\
3F7 & DSA(13) 2 bytes not used & F7EE \\
3F8 & DSA(14) 2 bytes & F7F0 \\
3F9 & DSA(15) 2 bytes & F7F2 \\
3FA & LAM PATTERN 2 bytes & F7F4 \\
3FB & & F7F6 \\
\hline
\end{tabular}

(RESERVED)

(\textit{DSA} stands for data storage address)

\textbf{Fig. 3} Interface memory map
Fig. 4 Interface to microprocessor
Fig. 5 CAMAC interface buffers
Fig. 7 Control line decoding from the microinstruction word
i) All signals are -ve logic enable.
ii) (..,) refers to the corresponding bits in the microinstruction word.
1. AN ITEL 8080 PROGRAM

The following example demonstrates the method of controlling the interface from an INTEL 8080 program. This is a subroutine invoked by the main program (running in the μP's memory) to initiate a single transfer of data between CAMAC and the interface memory.

Prior to calling this subroutine the following data declarations and parameter storing should have been performed.

a) DSA = address F7C0₁₆
b) ICA = address F800₁₆
c) STATUS = address F806₁₆
d) DATA = address where data for a CAMAC write command is stored or data obtained from a CAMAC read command will be stored.
e) NAF = address where the NAF is stored
f) N = address where the ICA data (ICA(N)) is stored.

; SINGLE TRANSFER MODE SUBROUTINE
;
STRM: LHLD NAF ; GET NAF COMMAND
XCHG ; SAVE IN D-E REGISTER PAIR
LDA N ; GET ICA DATA
RAL
RAL ; MULTIPLY BY 4 (i.e. DSA(1)=DSA(0)+4)
LXI H,DSA ; DSA(0) IN H-L
ADD L ; DSA(0)+4=N
MOV L,A ; H-L POINTS AT DSA(N)
MOV M,E ; STORE NAF LSB $DSA(N)
INX H
MOV M,D ; STORE NAF MSB $DSA(N)+1
INX H
SHLD TEMP ; SAVE DSA(N)+2 $TEMP
;
; READ, WAITE OR TEST? CAMAC COMMAND
;
MOV A,D ; GET NAF MSB IN A
ANI 01H ; F(16)=1?
JZ STRM1 ; NO THEN BRANCH
MOV A,E ; NAF LSB TO A
JP STRMW ; F(16)=1 & F(8)=0 THEN BRANCH
JMP STRMT ; F(16)=1 & F(8)=1 THEN JUMP
STRM1: MOV A,E  JP STRMR  ; F(16)=0 & F(8)=0 THEN BRANCH  FMP STRMT  ; F(16)=0 & F(8)=1 THEN JUMP  
  ; AUXILIARY WRITE (TRANSMIT) TO CAMAC  
  STRMw:  XCHG  ; SAVE DSA(N)+2 IN D-E  LHLD DATA  ; PUT DATA IN H-L  XCHG  ; MOV M,E  ; MOVE LSB TO DSA(N)+2  INXH  MOV M,D  ; MOVE MSB TO DSA(N)+3  CALL INITIATE  ; INITIATE CAMAC COMMAND  RET  
  ; AUXILIARY READ (RECEIVED) FROM CAMAC  
  STRMR:  CALL INITIATE  ; INITIATE CAMAC COMMAND  LHLD TEMP  ; GET DSA(N)+2 FROM TEMP  MOV E,M  ; READ LSB OF DATA INTO E  INX H  MOV D,M  ; READ MSB OF DATA INTO D  XCHG  SHLDDATA  ; PUT DATA INTO DATA  RET  
  ; AUXILIARY TEST (CAMAC)  
  STRMT:  CALL INITIATE  ; INITIATE CAMAC TEST COMMAND  RET  
  ; INITIATE CAMAC COMMAND STORED IN DSA(N)  
  INITIATE:  LDA N  ; GET ICA DATA  LXI H,ICA  ; MOV M,A  ; INITIATE COMMAND ICA(N)  E1  ; ENABLE INTERRUPT SYSTEM  HLT  ; WAIT FOR INTERRUPT  RET  ; RETURN  
  ; INTERRUPT ROUTINE FOR STRM  
  INTSTRM:  LDA STATUS  ; PUT STATUS WORD IN A RET  

A2. A TMS 9900 PROGRAM

*  
* SINGLE TRANSFER MODE SUBROUTINE  
*  
* STRM:  MOV $NAF,R6  ; CAMAC COMMAND IN R6  MOV $N,R7  ; N NUMBER IN R7  SLA R7,2  ; FORM 4*(N)  MOV R6,$DSA(R7)  ; STORE TO INTERFACE MEMORY  INCT R7  ; FORM NEXT ADDRESS (DSA+4(N)+1)
WHICH CAMAC COMMAND? (WRITE, READ, TEST?)

```
LI R8,010016 ; MARK FOR F(16)
COC R8,R6 ; TEST F(16)
JNE STRM1 ; F(16)=0 -> BRANCH
SRL R8,1 ; MARK FOR F(8)
COC R8,R6 ; TEST F(8)
JNE STRMW ; F(16)=1 & F(8)=0 -> WRITE
JMP STRMT ; F(16)=1 & F(8)=1 -> TEST
```

```
STRM1: SRL R8,1 ; MARK FOR F(8)
COC R8,R6 ; TEST F(8)
JNE STRMR ; F(16)=0 & F(8)=0 -> READ
```

* OTHERWISE: F(16) = 0 & F(8) = 1 -> TEST (AS WELL)

```
STRMT: BLWPaINIT ; INITIATE TEST
RTWP ; RETURN
```

```
STRMW: MOV aDATA,*R7 ; TRANSFER WRITE DATA TO INTERFACE MEMORY
BLWP @INIT ; INITIATE WRITE
RTWP ; RETURN
```

```
STRMR: BLWP @INIT ; EXECUTE CAMAC READ
MOV *R7,aDATA ; TRANSFER DATA TO μP MEMORY
RTWP ; RETURN
```

* INITIATION OF CAMAC COMMAND (SUBROUTINE)

```
INIT: MOV aN,aICA ; INITIATE COMMAND
IDLE ; WAIT FOR INTERRUPT
RTWP ; RETURN TO STRMX
```

* INTERRUPT ROUTINE (IS ENTERED FROM INIT)

```
INISTRM: MOV @STATUS,qSTATOUF ; SAVE STATUS OF INTERFACE
RTWP ; IN μP MEMORY (COUNTER) AND RETURN
APPENDIX B

MICROPROGRAMS

1. THE MICROPROGRAMS CALLED BY CAMAC

i) CAMAC WRITE (P(16)A(0))

MWRITE: CAM MEM(w) ; MODE W1-W16 INTO MEMORY @MAR
CAM MEM(w)
INCR MAR ; INCREMENT CONTENTS OF MAR
MOV MAR, MARB ; SAVE IN MARB
JMP 0 ; RETURN TO ADDRESS 0

ii) CAMAC READ (P(0)A(0))

MREAD: CAM RREG ; ENABLE THE CONTENTS OF RREG ONTO THE R1-R16 LINES
CAM RREG
CAM RREG
MOV MEM, RREG ; MOVE CONTENTS OF MEMORY @MAR INTO RREG
MOV MEM, RREG
INCR MAR ; INCREMENT CONTENTS OF MAR (ADDRESS)
MOV MAR, MARB ; SAVE IN MARB
JMP 0 ; RETURN TO ADDRESS 0

iii) Load MAR for READ (P(16)A(1))

WMAR1: CAM MAR ; MOVE W1-W16 INTO MAR (LOAD MAR FROM CAMAC)
CAM MAR
MOV MAR, MARB ; LOAD MARB
MOV MEM, RREG ; LOAD READ REGISTER
MOV MEM, REG
INCR MAR ; INCREMENT ADDRESS
MOV MAR, MARB ; LOAD MARB
JMP 0 ; RETURN TO ADDRESS 0

iv) Load MAR for WRITE (P(16)A(2))

WMAR2: CAM MAR ; LOAD MAR FROM CAMAC
CAM MAR
MOV MAR, MARB ; LOAD MARB
JMP 0 ; RETURN TO ADDRESS 0

v) Interrupt uP to inform it of the end of data transfer:

a) To memory

INT1: INT DTV
JMP 0

b) From memory

INT2: INT DTR
JMP 0
2. THE MICROPROGRAMS CALLED BY THE µP

2.1 Single transfer mode (STM)

Called at address FCO0 with data = 0-9.

STMO:  LD1 3E0,MAR;  JMP STM  ; LOAD MAR WITH DSA(0)

STMI:  LD1 3E2,MAR;  JMP STM

.  .  .

STM6:  LD1 3F0,MAR

STM:    JSR LNAF  ; CALL ROUTINE TO LOAD NAF REGISTER
        INCR MAR  ; INCREMENT ADDRESS
        MOV MAR,MARB  ; LOAD MARB
        JSR AUXCAMAC  ; CALL CAMAC ROUTINE

FINISH:  INT EOC  ; INFORM µP OF END OF CYCLE
         JMP 0  ; RETURN TO ADDRESS ZERO

; SUBROUTINE TO LOAD NAF REGISTER

LNAF:  MOV MAR,MARB  ; LOAD MARB
        MOV MEM,NAF  ; LOAD NAF
        RTS  ; RETURN

; SUBROUTINE TO PERFORM CAMAC OPERATION

AUXCAMAC:  BRIF AUXLOC,ERROR  ; TEST AUXILIARY LOCKOUT
            BRIF AUXREAD,AUXR  ; TEST FOR AUXILIARY READ
            BRIF AUXTEST,AUXT  ; TEST FOR AUXILIARY TEST

AUXW:  CAM MEM(T)  ; START DATAWAY CYCLE
        CAM MEM(T) AND BRIF AUXLOC,AUXCAMAC  ; TRANSMIT DATA TO CAMAC AND
        CAM MEM(T),S1  ; READ FROM MEMORY
        CAM MEM(T),S2
        CAM MEM(T)
        RTS

AUXR:  CAM MEM(R)  ; RECEIVE DATA FROM CAMAC AND
        CAM MEM(R) AND BRIF AUXLOC, AUXCAMAC; WRITE TO MEMORY
        CAM MEM(R), S1
        CAM NAF,S2
        CAM NAF
        RTS

AUXT:  CAM NAF  ; AUXILIARY TEST CAMAC COMMAND
        CAM NAF, AND BRIF AUXLOC, AUXCAMAC; ENABLES NAF REGISTER ONTO CAMAC
        CAM NAF,S1
        CAM NAF,S2
        CAM NAF
        RTS

ERROR:  INT AUXLOC
         JMP 0
2.2 ZCI initiation

ZCI:  LD1 3F2, MAR
     BRIF AUXLOC, ERROR
     CAM ZC1
     CAM ZC1
     CAM ZC1, S1 AND BRIF AUXLOC, ERROR
     CAM ZC1, S2
     CAM ZC1
     JMP 0

2.3 Load counter (ICA(10))

LOADCNTR:  LD1 3F4, MAR
              MOV MAR, MARB
              MOV MEM, CNTR
              MOV MEM, CNTR
              JMP 0

2.4 Load MAR (ICA(11))

LOADMAR:  LD1 3F5, MAR
              MOV MAR, MARB
              MOV MEM, MAR
              MOV MEM, MAR
              JMP 0

2.5 Load NAF (ICA(12))

LOADNAF:  LD1 3F6, MAR
              MOV MAR, MARB
              MOV MEM, NAF
              MOV MEM, NAF
              JMP 0

2.6 Block transfer mode (ICA(13))

This microprogram assumes that the NAF, MAR and CNTR have been loaded prior to it being called. The loading of the MAR must be the last operation initiated by the μP before calling the block transfer mode microprogram.

BTRM:  JSR AUXCAMAC
        BRIF X, OK

NOTX:  LD1 3F4, MAR
        MOV CNTR, MEM
        MOV CNTR, MEM
        INT NX
        JMP 0

OK:  BRIF Q, CONTINUE

NOTQ:  LD1 3F4, MAR
        MOV CNTR, MEM
        MOV CNTR, MEM
        INT NQ
CONT:  JMP  0
       DECR  CNTR
       INCR  MAR
       BRIF  CNTR<Z,FINISH
       JMP  RTRM

2.7 'A' scan mode ICA(14)

This microprogram also assumes that the MAR and NAF command register have
been loaded prior to it being called

ASCAN:  JSR  AUXCAMAC
       BRIF  Q,AOK
       JMP  NOTQ
       INCR  A
       INCR  MAR
       JMP  FINISH

2.8 Demand message initiation (ICA(15))

DMI:  BRIF  DBSY,DMIERR
      LD1  3F7,MAR
      MOV  MEM,SGLE
      MOV  MEM,SGLE
      JMP  FINISH

DMIERR:  INT  DBSY
          JMP  0

3. LAM MONITOR ROUTINE

This microprogram resides at address zero and is repeated continuously by
the interface controller when it has no other task to perform

LAMR:  LD1  3F8,MAR
       MOV  LAM,MEM
       MOV  LAM,MEM
       BRIF  LAM,LAMPRINT
       JMP  LAMR

LAMPRINT:  INT  LAM

HERE:  JMP  HERE

4. MICROPROGRAM MEMORY MAP

<table>
<thead>
<tr>
<th>Start address</th>
<th>End address</th>
<th>Microprogram name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>06</td>
<td>LAMR</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>0B</td>
<td>MWRITE</td>
<td></td>
</tr>
<tr>
<td>0C</td>
<td>13</td>
<td>MREAD</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1B</td>
<td>WMAR1</td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>1F</td>
<td>WMAR2</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>INT1</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>INT2</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td>STM0</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>STM1</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>STM2</td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>2B</td>
<td>STM3</td>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>2D</td>
<td>STM4</td>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>2F</td>
<td>STM5</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>STM6</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>STM7</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>STM8</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>37</td>
<td>STM9</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>3D</td>
<td>STM</td>
<td></td>
</tr>
<tr>
<td>3E</td>
<td>41</td>
<td>LNAF</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>56</td>
<td>AUXCAMAC</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>58</td>
<td>LOADCNTR</td>
<td></td>
</tr>
<tr>
<td>5C</td>
<td>60</td>
<td>LOADMAR</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>65</td>
<td>LOADNAF</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>76</td>
<td>BTRM</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>70</td>
<td>ASCAN</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>84</td>
<td>DM1</td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>E0</td>
<td>ICA(0)</td>
<td>JMP STM0</td>
</tr>
<tr>
<td>E1</td>
<td>E1</td>
<td>ICA(1)</td>
<td>JMP STM1</td>
</tr>
<tr>
<td>E2</td>
<td>E2</td>
<td>ICA(2)</td>
<td>JMP STM2</td>
</tr>
<tr>
<td>E3</td>
<td>E3</td>
<td>ICA(3)</td>
<td>JMP STM3</td>
</tr>
<tr>
<td>E4</td>
<td>E4</td>
<td>ICA(4)</td>
<td>JMP STM4</td>
</tr>
<tr>
<td>E5</td>
<td>E5</td>
<td>ICA(5)</td>
<td>JMP STM5</td>
</tr>
<tr>
<td>E6</td>
<td>E6</td>
<td>ICA(6)</td>
<td>JMP STM6</td>
</tr>
<tr>
<td>E7</td>
<td>E7</td>
<td>ICA(7)</td>
<td>JMP STM7</td>
</tr>
<tr>
<td>E8</td>
<td>E8</td>
<td>ICA(8)</td>
<td>JMP STM8</td>
</tr>
<tr>
<td>E9</td>
<td>E9</td>
<td>ICA(9)</td>
<td>JMP STM9</td>
</tr>
<tr>
<td>EA</td>
<td>EA</td>
<td>ICA(10)</td>
<td>JMP LOADCNTR</td>
</tr>
<tr>
<td>EB</td>
<td>EB</td>
<td>ICA(11)</td>
<td>JMP LOADMAR</td>
</tr>
<tr>
<td>EC</td>
<td>EC</td>
<td>ICA(12)</td>
<td>JMP LOADNAF</td>
</tr>
<tr>
<td>ED</td>
<td>ED</td>
<td>ICA(13)</td>
<td>JMP BTRM</td>
</tr>
<tr>
<td>EE</td>
<td>EE</td>
<td>ICA(14)</td>
<td>JMP ASCAN</td>
</tr>
<tr>
<td>EF</td>
<td>EF</td>
<td>ICA(15)</td>
<td>JMP DM1</td>
</tr>
<tr>
<td>F0</td>
<td>F0</td>
<td>F(0)A(0)</td>
<td>JMP MREAD</td>
</tr>
<tr>
<td>F4</td>
<td>F4</td>
<td>F(16)A(0)</td>
<td>JMP MWRITE</td>
</tr>
<tr>
<td>F5</td>
<td>F5</td>
<td>F(16)A(1)</td>
<td>JMP WMAR1</td>
</tr>
<tr>
<td>FA</td>
<td>FA</td>
<td>F(24)A(0)</td>
<td>JMP WMA</td>
</tr>
<tr>
<td>FB</td>
<td>FB</td>
<td>F(24)A(1)</td>
<td></td>
</tr>
</tbody>
</table>
PARALLEL BRANCH HIGHWAY SYSTEM
(STAR/CAMAC CRATE CONTROLLER AND TYPE A2 CONTROLLERS)

Only in the STAR/CAMAC crate controllers and the Type A2 crate controllers does an auxiliary controller have access to the L and N lines, via a back panel connector. There is, however, no means for the µP to transmit a message to the central computer, except by a LAM (i.e. the individual LAM Line of the interface module), and no prior warning to an impending crate controller dataway cycle is available. It is therefore more difficult for an 'intelligent' auxiliary crate controller to time share the dataway in a parallel crate than in a serial crate system.

CRATE BUSY SIGNAL (B)

The B signal is obtained directly from the CAMAC dataway and is enabled during a crate controller CAMAC command. On this signal becoming enabled the interface module’s CAMAC buffers are disabled to prevent possible conflicts in dataway signals. Any auxiliary CAMAC command is aborted and the µP informed by means of an interrupt and the B (in replace of DBSY bit 6, see Section 4.4) bit in the status word being set.

It is now the responsibility of the µP to decide whether or not the auxiliary CAMAC command need be repeated. If the command had reached S2 timing before being aborted then it need not be repeated.