CAMAC Interface for the ESOP Microprocessor

Abstract

A logic unit is described, which connects the ESOP \([1,2]\) fast microprogrammable processor to a CAMAC dataway. Through this interface programs and data can be loaded, routines started at any location and results read back.

A prototype has been built and tested, using wirewrap technique. A printed circuit version is being considered.

A FORTRAN utility routine has been written and tested, which executes the CAMAC functions found in Table 1. Further, a microprogram has been implemented to execute the block transfer operations described in chapter 2.4.
Index

1. Introduction 1
2. Description of the Hardware
   2.1 Physical Lay-out 1
   2.2 Interface to the Data Memory 1
   2.3 Interface to the Instruction Memory 3
   2.4 Block Transfer Functions 5
   2.5 The LAM function 8
References 9
Figure Captions 9
Fig. 1. CAMAC Interface 10
Fig. 2. IM Interface and command 11
Fig. 3. Programmed data transfer from 'ESOP' to CAMAC dataway: Timing diagram 12
Table 1. CAMAC Instructions 13/14
Table 2. Microprocessor I/O functions 15
1. Introduction

The ESOP $^{[1,2]}$ Microprocessor was originally designed to be connected to the PDP-11 UNIBUS via the ERASME digitizing logic interface. In particular, the data memory (DM) was assigned 512 byte addresses in the PDP-11 address space, which means that it looks like part of the computer's own memory.

The UNIBUS does not have any "Function" lines which can be compared with the CAMAC "F" signals. Instead, certain addresses are interpreted as commands. Typical examples are master clear "mn" and the pseudo instructions M.EXEC, M.COM etc.

Whereas CAMAC use "Function" signals, there are no address lines as such in the system. The "Sub-address" lines A1, A2, A4 and A8 cannot be used to address a memory of reasonable size. Therefore, indirect addressing must be used when transferring data through CAMAC to or from the host computer.

Finally, the interfacing protocol of the two systems are based on different philosophies: PDP-11 uses a handshake system (MSYN-SSYN) whereas a CAMAC cycle is fixed in time. However, although the cycle is rigidly specified, signals can only be interpreted by the strobes S1 and S2. For multi-cycle operation, such as a data block-transfer, there is no "legal" way to find out when a cycle starts and when it ends. Thus, edge-triggering on a NAF or B signal is illegal, because proper operation depends upon the crate controller design.

In the case where a data transfer cycle has to be prolonged, pending upon some condition, the SSYN can simply be delayed in the case of the PDP-11 system. CAMAC has no such facility, but at CERN an auxiliary "HOLD" signal has been assigned to P2. This feature is used in the present interface.

2. Description of the Hardware

2.1 Physical Lay-out

The microprocessor is built up from a minimum of three plug-in units. These units are CAMAC type boards, and they lodge in a CAMAC type chassis. They do not obey the CAMAC standards as far as the DATAWAY is concerned. The backboard is a special one, and the power-supply is +5V.

The microprocessor-CAMAC interface consists of three plug-in units connected by two cables. Unit MPC1 (CAMAC interface unit) is plugged into any slot of a CAMAC chassis and the other units lodge in the microprocessor chassis. The latter two mainly contain drivers and receivers. In the final version, only one plug-in unit with transmitter-receivers will be needed in the microprocessor chassis.

2.2 Interface to the Data Memory (DM)

2.2.1 Principle

The logic diagram of the interface is shown on Fig. 1. We shall start by explaining the interface to the data memory (DM).
(A block diagram of the DM is found in ref. 2, and a
detailed schematic on CERN-DD drawing S-059-331A-1)

Two control signals are used to read from or write to the
DM: SEL-L (pin 28 of unit DM) and IN-H (pin 20 of unit DM)
when the latter signal is low, it is interpreted as WRITE.

The data comes in through a set of tristate gates connected
to the Data Bus (DB <15-0>) which is under control of the
"data-bus gating" logic. Output from the memory pass through
UNIBUS driver gates from the DB to the CAMAC R-line-drivers.

In the micro-processor the address lines <5-0> are used as
well for DM addressing as functions. Therefore, a 2:1 multi-
plexer (2 x 74S257) is needed in the interface, which will
select either a set of functions (M.IM, M.EXEC, M.IAR or
M.COM) or the output of the address register CDA.

2.2.2 Transfer of data from CAMAC to the microprocessor DM

The transfer takes place in two steps:

1. The relevant address is loaded into the CDA register by
   N.A(1).F(17).S1. The multiplexer is connected to CDA
   because all the M. functions (M.IM etc.) are 0.

2. The data is transferred to the DM via the data bus by

Note that two signals are involved: IN-H which enables the
memory WRITE, and SEL-L = N.A(6).F(16).S1 which generates a
synchronized strobe.* (Schematic S-59-331A-1 "DATA MEMORY".)

This mode of transfer, using the CDA register, needs two
CAMAC cycles per word, and is best suited for random access.
Block transfer can be carried out as outlined in a separate
chapter 2.4.

2.2.3 Transfer of data from the microprocessors DM to CAMAC:

The transfer takes place in two steps:

1. Set up the address in CDA by means of N.A(1).F(17).S1.
2. Fetch the addressed word from the DM by means of N.A(6).F(0).

The latter function generates a static SELECT signal which

---

* [Due to the internal connection between the WRITE/READ, ENABLE and
the output gates of the memory chips SN74200, the strobe is given on
the ENABLE input rather than on the WRITE, keeping in mind that the
bi-directional data bus is connected to the input as well as the
output of the DM. Care must be taken when replacing the SN74200
by other manufacturers' chips, because their internal logic might be
slightly different. In particular the SIGNETICS 82S16 does not
work in the DM, but is OK for the I.M].
enables the DM output.

The IN-H function is now LOW, because F(16) = 0 by exclusion. In other words, the DM is always in READ state unless a specific WRITE is given.

The N·A(6)·F(0) function will also enable the read gates connected to R<16-1> of the HIGHWAY. The time it takes from the recognition of the read function until the data is read out and set up on the R lines is in the order of 100-200 ns, yielding ample margin in front of the S1 pulse in the CAMAC crate controller.

2.3 Interface to the Instruction Memory (IM)

2.3.1 Principle

A block diagram of the instruction memory (IM) is found in ref. 2. The schematic diagram has the DD reference number S-59-383-1.

Normally the IM is first loaded, and then the content read back and checked. The subsequent 16 bit words from the CAMAC control computer are automatically assembled into 48 bit instructions in the IM by means of a shift register sequencer. When connected to the "EDI" interface of the PDP-11, data transfer takes place by means of two signals: EN02, which is used for read, and RI2 which is used for write. These signals are derived by decoding the UNIBUS address lines and AND'ing with the CO, Ci lines as well as the MSYN.

Due to the differences in the interfacing protocol between PDP-11 and CAMAC, as outlined in chapter 1, two additional signals had to be brought in for the CAMAC case: OUTPUT MUX, EN and MEM EN-H. (See diagram S-59-355-2). These signals are connected by means of jumpers on the board to a CAMAC or PDP-11 position.

As already mentioned in chapter 1, the PDP-11 does not use "Function" lines.

CAMAC uses functions which are to a large extent standardized. The easiest way to translate CAMAC functions into microprocessor commands was to encode the individual F outputs of the F decoder (which is needed to generate the X and Q responses anyhow) and pass them on through the 2 : 1 address multiplexer. To switch the multiplexer to the relevant input, all valid functions were OR-ed together so that whenever any of the functions is true, the function input (B) is selected.

Apart from the two jumpers mentioned, no other modifications in the microprocessor are necessary to change from PDP-11 to CAMAC operation.
2.3.2 The Instruction Memory Function Set

The procedure to load the instruction memory is exactly the same as for the case of the PDP-11. The relevant routines are found in ref. 6.

Although the instructions look the same, some additional features had to be implemented in the CAMAC interface in order to make it compatible with the input logic of the IM. The microprocessors IM interfacing logic is shown in Fig. 2: IM INTERFACE AND COMMAND.

Each CAMAC function will generate one or more signals to the IM logic as described below:

a) \( N \cdot A(2) \cdot F(17) = M \cdot COM \) will generate a strobe at time \( S_1 \) on the \( R12-L \) line, and thus load the incoming word on the \( W < 7-1 \) lines into the EXT COMMAND register. (See schematic diagram S-059-355-2).

Note that the \( R12-L \) strobe is generated whenever \( F(17) \) AND any of the \( M \) functions except \( A(4) \) are true corresponding to the first item of the following expression

\[
R12-L = S_1 \cdot \{A(4) \cdot F(17) \cdot [A(2) \cdot A(3) \cdot A(4) + A(5)] \cdot A(4) \cdot F(16)\}
\]

The internal IM control code for \( M \cdot COM \) is 76.

b) \( N \cdot A(3) \cdot F(17) = M \cdot IAR \) generates code 56 in the IM interface. (See Fig. 4). A strobe on the \( R12-L \) line will occur at time \( S_1 \) (\( G_1, G_2, G_3 \) on Fig. 1). This strobe is AND'ed with the \( M \cdot IAR \) level in gate 3 on Fig. 2, the output of which will pass on to the Instruction Address Register's (IAR) clock input. The effect is that the data on \( W < 8-1 \) will be loaded into the IAR.

There is one side effect: the sequencer will be cleared, provided that the relevant command bit has been set to LD ("Load").

c) \( N \cdot A(4) \cdot F(16) = M \cdot IM(WRITE) \)

This function writes into the IM at the current address set up in the IAR. The IM control code is 66.

Side effect: advances sequencer and advances (IAR) every third 16 bit word, provided the relevant command bits have been set previously. The data on the \( W < 16-1 \) lines are strobed into the IM by means of the gated \( S_1 \) pulse (\( G_4, G_5, G_6 \) on the MEM EN-H line). In addition, \( R12-L \) is pulsed to advance the sequencer (gates 6, 5, 2 on Fig. 2) and strobe the IAR each time it is in the Qc position. This means that three 16 bit words are automatically assembled into one 48 bit word and the instruction address incremented after every third transfer.

The command bits IW and SH must be set up prior to the transfer. (See Schematic: Instruction Memory Control Logic: S-059-352-1).
d) \( N.A(4).F(0) = M.IM \) (read)

This function is used to read the IM for checking.

Side effect: Advances the sequencer in order to select the correct 16 bit portion of a 48 bit instruction word and strobe IAR every third time.

Note that the least significant 16 bit word is read via the data bus DB, and only the two most significant ones through the 2:1 output multiplexer shown in Fig. 2. This saves hardware, and the procedure is not time critical.

The IM control code is the same as for the previous function (66).

The data is enabled statically on to the R<16-1> lines by means of the MEM EN-H signal (G7, G6 on Fig.1). The sequencer is post advanced at time S2 through the EN02-L line.

e) \( N.A(5).F(17) = M.EXEC \)

This function will start execution of a micro-routine at the absolute address given on W<8-1>.

Side effects: None.

The program must of course have been loaded, and the IM-command register properly set up. The IM control code is 46.

The function simply generates a RI2-L pulse at time S1. This pulse is synchronized in the microprocessor, and the starting address is presented on to the Instruction Address Bus (IAB) for one cycle. At the end of the cycle the starting address is transferred to the IAR.

The transfer is done under program control: that is a M.EXEC routine must be present and active in the processor. (See ref.3, 5).

2.4 Block Transfer Functions

2.4.1 Principle

In chapter 2.2 it was described how to read or write from the data memory (DM) using indirect addressing. The DM had to be initiated by a micro-routine to put it in "EXTERNAL" mode, and the clock had to be running, but no program had to be executed during the transfer. In fact, with some additional hardware to hold the relevant instruction bits, a DM could work entirely on its own.
Due to the indirect addressing, the transfer rate is one word per two CAMAC cycles. In addition, the wordcount has to be set up and checked externally.

The microprocessor is of course capable of carrying out a programmed data transfer, and as will be shown below, the speed of operation is so high, that there is ample time to do the addressing and wordcount internally within a CAMAC cycle.

2.4.2 Block Transfer Procedure

We suppose that the relevant block-transfer routines[7] are residing in the instruction memory IM. In addition, a "flag"-check routine checks for incoming flags once per \( \mu \)-cycle. (Quasi-interrupt routine). (See ref. 1, 2, 3). All we have to do, is to set a flag by means of a 250 ns wide pulse, and the corresponding routine will start. In the case of a read function the program in the micro-processor will fetch a word from the IM, place it on the data bus (DB) and update address and wordcount for the next transfer.

In the case of a write function, the word will be gated on to the data bus DB written into the memory, and then the address and wordcount incremented.

The programs end with a return to the flag-check routine.

The transfers have of course to be initiated: The starting address in the IM must be set up in the DAR and the number of words to be transferred in the wordcounter. In the case of a read function the initiating routine also fetches the first word from the IM and puts it on to the DB, ready for transfer to CAMAC. The initiating routines are called by means of a M.EXEC prior to each block transfer. The starting address and wordcount can be changed by reloading the corresponding instructions in the IM.

2.4.3 Hardware Description

There are two different read functions and one write function (See Fig. 1 and the timing diagram Fig. 3.) The simplest read function is:

\[ N \cdot A(7) \cdot F(1) \]

which merely enables the common readout gates and then sends a "FLAG" signal to the microprocessor (G9, G10). The processor detects the 'FLAG' and starts a read routine (which must be written to fit the specific task). The routine works one step ahead of the read command: the word to be transferred to the dataway is fetched from the IM in the previous CAMAC cycle, and is waiting on the Data Bus. The R lines will not change before the beginning of S2 at the earliest.

The Q-reply is under micro-program control.
It will be set by the initiating routine, and then turned off after the last word has been transferred. In order to prevent CAMAC hang-up, the Q flip-flop can be cleared by a separate function:

\[ N \cdot A(7) \cdot F(24) \]

It is of course also cleared by the Z-signal.

As shown on the timing diagram Fig. 3, it is possible to do some simple operations on each word before it is transferred to the CAMAC system. For instance, one could add a constant to it and/or check the sign.

For longer routines, the HOLD function has been introduced. The "read and hold" has a separate code:

\[ N \cdot A(8) \cdot F(1) \]

The output of the gate G11 will immediately generate the HOLD signal on P2, which inhibits S1 and S2 in the crate controller. The FLREQ flip-flop is set causing a quasi-interrupt in the microprocessor.

At the end of the micro-routine, the ENHLD and the FLREQ flip-flops are reset, and the HOLD signal drops. Now S1 and S2 are generated, at S1 the data from the microprocessor is read, and at S2 the ENHLD is set again.

A complicated situation now occurs, which is due to the way the CAMAC cycle is defined (or not defined): there is no way to determine when the present cycle ends and when the next one starts. Normally A(8) . F(1) are still present at time S2. Therefore FLREQ is turned on again before the next CAMAC command has arrived. It will, however, only be reset once per cycle provided the micro-routine is not too short. The programmer therefore has to time his program in such a way that it fits the controller (and computer) he is using.

As the wordcount is done by the microprocessor, the transfer can be correctly terminated in spite of the difficulties mentioned above.

To write a block of words from CAMAC to the micro-processor the function

\[ N \cdot A(7) \cdot F(16) \]

can be used. This function triggers a "FLAG" and starts a micro-routine which fetches the word from the CAMAC W bus and writes it into the data memory (DM). The wordcount and address are updated, and then the micro-routine returns to FLAGCHECK (normally in loc. 0 of the instruction memory).

Upon completion of the transfer a LAM can be sent by means of a MARKER.
2.5 The LAM function

The LAM logic has been designed in accordance with the specifications
given in the "CAMAC bulletin" No. 6, March 1973 Supplement. The L-STAT
is set by means of a "MARKER" bit in the micro-processors instruction
word.

The CAMAC functions are:

\[
\begin{align*}
N\cdot A(0) \cdot F(26) & \text{; enable LAM MASK.} \\
N\cdot A(0) \cdot F(24) & \text{; disable LAM MASK.} \\
N\cdot A(0) \cdot F(27) & \text{; test L-STATUS.} \\
N\cdot A(0) \cdot F(10) & \text{; clear L-STATUS.}
\end{align*}
\]

The L-STAT is of course also cleared by Z.S2.

The L-STAT can be set by means of a micro-instruction: MARKER, PI,
where PI, the MARKER number, has to be determined when wiring the inter-
face. (See TABLE 2).

Acknowledgements

I want to express my gratitude to Mr. Bruno Gaultier who on leave
from the Ecole Nationale de l'Aviation Civile, Toulouse, built and
carried out the initial testing of the prototype of the unit described
in this report.

I also want to thank Mr. Robert McLaren who assisted Mr. Gaultier
and took over the unit for further testing when he left.
References


3. D. Myers, MICRO DOC.

4. D. Myers, PALXM DOC.

5. D. Myers, EHT 10 LIS (Listing).


7. T. Lingjaerde, "CAMBLOCK" (Microprogram listing).

---

Schematic diagrams referred to in this report

<table>
<thead>
<tr>
<th>CERN DD Drawing No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-059-189-2</td>
<td>ERASME EDI: UNI-to EDI- bus interface</td>
</tr>
<tr>
<td>S-059-190-2</td>
<td>ERASME EDI: DMA interface</td>
</tr>
<tr>
<td>S-059-331A-1</td>
<td>ESOP data memory</td>
</tr>
<tr>
<td>S-059-335-2</td>
<td>ESOP instruction memory interface and command</td>
</tr>
<tr>
<td>S-059-383-1</td>
<td>ESOP instruction memory</td>
</tr>
</tbody>
</table>
Fig. 3: Programmed data transfer from 'ESOP' to CAMAC dataway: Timing diagram
<table>
<thead>
<tr>
<th>No.</th>
<th>MNEM.</th>
<th>Operation</th>
<th>CAMAC COMMAND</th>
<th>Comment</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>W&lt;7-1-&gt;CDA</td>
<td>N·A(1)·F(17)</td>
<td>Indirect addr. setup</td>
<td>2.2.2.-1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>W&lt;16-1-&gt;DM&lt;150&gt;</td>
<td>N·A(6)·F(16)</td>
<td>DMA transfer to DM must be preceded by 1</td>
<td>2.2.2.-1</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>DM&lt;15-0-&gt;W&lt;16-1&gt;</td>
<td>N·A(6)·F(0)</td>
<td>Transfer data from DM to CAMAC R-bus. Must be preceded by 1</td>
<td>2.2.3.-2</td>
</tr>
<tr>
<td>4</td>
<td>M.COM</td>
<td>W&lt;8-1-&gt;COMM.REG</td>
<td>N·A(2)·F(17)</td>
<td>To set up command reg. in ESOP</td>
<td>2.3.2.-a</td>
</tr>
<tr>
<td>5</td>
<td>M.IAR</td>
<td>W&lt;8-1-&gt;IAR</td>
<td>N·A(3)·F(17)</td>
<td>Set up instruction address in ESOP</td>
<td>2.3.2.-b</td>
</tr>
<tr>
<td>6</td>
<td>M·IM(WRITE)</td>
<td>W&lt;16-1-&gt;IM</td>
<td>N·A(4)·F(16)</td>
<td>Load instruction memory. Must be initially preceded by 5</td>
<td>2.3.2.-c</td>
</tr>
<tr>
<td>7</td>
<td>M·IM(READ)</td>
<td>IM &lt;16bits-&gt;R 16-1</td>
<td>N·A(4)·F(0)</td>
<td>Read instr. memory</td>
<td>2.3.2.-d</td>
</tr>
<tr>
<td>8</td>
<td>M·EXEC</td>
<td>W&lt;8-1-&gt;IAR; START EXECUTION</td>
<td>N·A(5)·F(17)</td>
<td>Start routine at addr. W&lt;8-1&gt;</td>
<td>2.3.2.-e</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>SET FLAG 7; DB&lt;15-0-&gt;R&lt;16-1&gt;</td>
<td>N·A(7)·F(1)</td>
<td>Block transfer read using micro-routine started by FLAG 7</td>
<td>2.4.3</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>SET FLAG 6; HOLD (P2)</td>
<td>N·A(8)·F(1)</td>
<td>Same as 9 but using CAMAC 'HOLD' function (P2). Works only for crate-controller with HOLD!</td>
<td>2.4.3</td>
</tr>
<tr>
<td>No.</td>
<td>MNEM.</td>
<td>Operation</td>
<td>CAMAC COMMAND</td>
<td>Comment</td>
<td>Reference</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------</td>
<td>-------------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>SET FLAG 5; W&lt;16-1&gt;-&gt;DB&lt;15-0&gt;</td>
<td>N.A(7).F(16)</td>
<td>Block transfer write to DM under micro program started by FLAG 5.</td>
<td>2.4.3</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Clear Q-line</td>
<td>N.A(7).F(24)</td>
<td>Clear Q-line used in case of abort during block transfer</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>ENABLE LAM</td>
<td>N.A(0).F(26)</td>
<td></td>
<td>2.5</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>DISABLE LAM</td>
<td>N.A(0).F(24)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>TEST LAM</td>
<td>N.A(0).F(27)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNCTION</td>
<td>MICROPROCESSOR I/O functions. (&quot;FLAGS&quot; and &quot;MARKERS&quot;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLAG 6</td>
<td>Used to trigger a read operation in block transfer mode from the processor to CAMAC, in HOLD [CAMAC (P2)] mode. The user must determine which priority he wants and connect this line to the relevant FLAG input of the processor. (See N A(8) F(1) on table 1.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLAG 7</td>
<td>Same as above, but without HOLD on the CAMAC (P2).</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARKER 1</td>
<td>Signal from the micro-processor used to set the Q flip-flop to indicate start of block-transfer. The relevant micro-instruction must be issued as a result of a CNAF which itself sets the Q at the beginning of the CAMAC cycle, and the MARKER given before the end. In other words, the QFF is used to hold Q during a block transfer.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARKER 2</td>
<td>Is used to clear the Q-FF at the end of a block transfer.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARKER 3</td>
<td>Clears FLAG 6 REQ. and the ENHLD to release HOLD (P2) for a prolonged CAMAC cycle.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARKER 4</td>
<td>Sets L-STAT, which in turn generates a LAM provided L-MASK has been set.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>