LA-7683-MS
Informal Report

Special Distribution
Issued: February 1979

Design and Operation of the LAMPF Auxiliary Controller

High-Speed Remote Processing on the CAMAC Dataway

Donald R. Machen

[illegible]

CONTENTS AND PRINT REFERENCE

*Prototype documentation available from the LAMPF Drawing Office under 63Y-182693, Sheets 1-28.

FIGURES

DESIGN AND OPERATION OF THE LAMPF AUXILIARY CONTROLLER

High-Speed Remote Processing on the CAMAC Dataway

by

Donald R. Machen

ABSTRACT

A CAMAC Auxiliary Controller has been developed to further the concepts of distributed processing in both process control and experiment data-acquisition systems. The Auxiliary Controller is built around a commercially available 16-bit microcomputer and a high-speed bit-sliced microprocessor capable of instruction execution times of 140 ns. The modular nature of the controller allows the user to tailor the controller capabilities to the system problem, while maintaining the interface techniques of the CAMAC Standard.

I. INTRODUCTION

This report covers both the design details and operation of an LSI-11/2* CAMAC Auxiliary Crate Controller (ACC). I will also attempt to convey how such a device can be used with CAMAC data and control systems and what options may be taker with respect to the several modules that comprise the ACC.

The Auxiliary Controller discussed in this report is based upon a design done around the Ferranti F100L (a 16-bit single-chip microprocessor) that is presently operational at the Daresbury Laboratory in England. The concept has remained the same between the two ACCs; only the necessary hardware details have changed to accommodate the LSI-11/2 "Q-Bus."

In order to fully understand the contents of this report, the reader must have a reasonable grounding in the reference documents listed below. Without some knowledge of the CAMAC standard, or the material published by DEC and Advanced Micro Devices (AMD), the technical details of the ACC may become somewhat opaque.

*Digital Equipment Corporation (DEC) trademark

Reference Documents

1. IEEE Standard 583-1975 (EUR-4100e), "Modular Instrumentation and Digital Interface System - CAMAC"

2. IEEE Standard 595-1976 (EUR-6100e), "Serial Highway Interface System - CAMAC"

3. IEEE Standard 596-1976 (EUR-4600e), "Parallel Highway Interface System - CAMAC"

4. DOE/EV-0007 (EUR-6500e), "Multiple Controllers in a CAMAC Crate"

5. TID-26618, "CAMAC Tutorial Articles," USDOE Publication Covering All Aspects of the CAMAC System

6. Digital Equipment Corporation, "1978 Microcomputer Handbook and Addenda Covering the LSI-11/2"

7. Advanced Micro Devices, Inc., AMD-2900, "Bipolar Microprocessor Family Data Book"

8. D. R. Machen, C. G. Ratcliffe, and A. C. Peatfield, "A Multi-Microprocessor Auxiliary Crate Controller for Front-End Data Processing in CAMAC," Daresbury Laboratory/Science Research Council, England, report DL/CSE/77-1,

   and

   IEEE Transactions on Nuclear Science, Vol. NS 25, No. 1, p 711, February 1978.

Reference documents 1 to 3 describe fully the CAMAC system of instrumentation, while Ref. 4 explains how multiple auxiliary controllers may share the Dataway of a single CAMAC crate. Reference 5 contains an interesting and informative set of papers on CAMAC. It is recommended reading for anybody attempting to plow through the formal CAMAC specifications. References 6 and 7 are essential to the understanding of the ACC described in this report, while ref. 8 discusses the ACC based on the Ferranti microprocessor.

The reader, at this point, should turn to Appendix A of this report in order to review the initial system design concepts upon which the LAMPF* ACC is based. An understanding of the ideas behind front-end and parallel processing in a CAMAC instrumentation environment will aid in understanding the details that follow in this report.

System Configuration

The CAMAC system configuration shown in Fig. 1 indicates where, in a large CAMAC system, multiple auxiliary controllers can be included. The motivating factor for auxiliary controllers is two-fold: Firstly, remote parallel processing can be accomplished in CAMAC crates interfaced to a larger host computer, and secondly, high-speed, front-end processing, or filtering, can be realized in data collection systems. The first case is generally useful in distributed control systems, wherease the second case finds use in experiments (in various fields) where data rates are high and not all the data are considered "good." Further, the overall gains possible in system throughput and response to remote stimuli are also motivating factors.

*LAMPF: The Clinton P. Anderson Meson Physics Facility

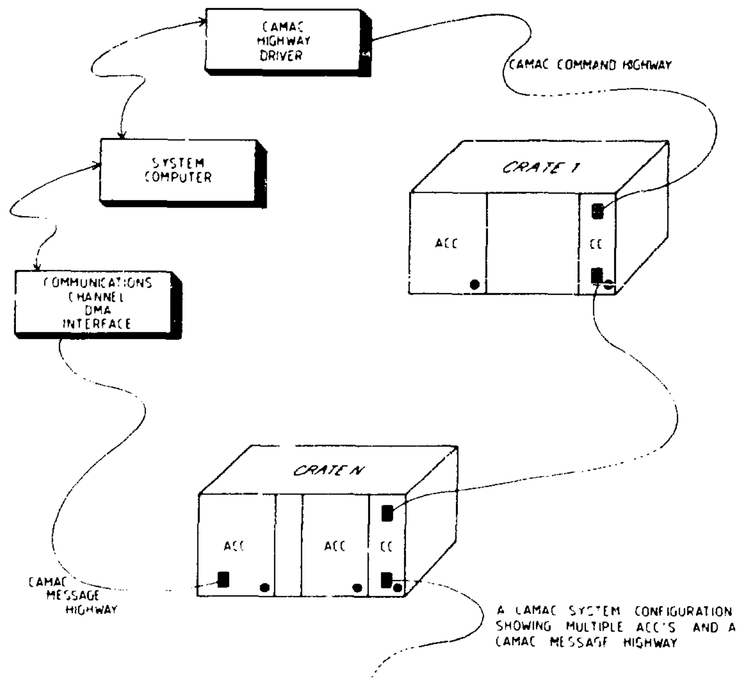

CAMAC HIGHWAY DRIVER
CAMAC COMMAND HIGHWAY
SYSTEM COMPUTER
CRATE 1
ACC
CC
COMMUNICATIONS CHANNEL DMA INTERFACE
CRATE N
ACC
ACC
CC
CAMAC MESSAGE HIGHWAY
A CAMAC SYSTEM CONFIGURATION SHOWING MULTIPLE ACC'S AND A CAMAC MESSAGE HIGHWAY

Fig. 1. System configuration.

Although not physically provided with either the Daresbury or the LAMPF prototype versions of the Auxiliary Controller, a high-speed (2-MHz) bit-serial, message-oriented block-transfer channel can easily be added to complement the well-known single-action CAMAC highways. Such an autonomous message channel can greatly enhance the data transfer rate and throughput of a front-end processor ACC.

A practical limit to the number of ACCs per crate is two to three. Although DOE/EV-0007 (EUR-6500e) allows eight ACCs, one soon realizes that space and power constraints prevail. The spectre of a crate full of ACCs, all talking to one another and to the system computer, may be an interesting exercise but has little value in the practical world of data acquisition and process control.

ACC Configuration

The LAMPF ACC is modular and flexible from the point of view that not all components are needed in order to configure a working system. Figure 2 shows the basic idea--that of a microprocessor-implemented controller, with Control Port, Dataway Access Port, processor memory, and a fast firmware subroutine facility. The Control Port provides the ACC access to all CAMAC Dataway signal lines with the exception of station number lines and LAM lines. The latter two are sent and received, respectively, via the Auxiliary Controller Bus (ACB). Further, priority arbitration for Dataway "mastership" takes place along the ACB (the reader is referred to DOE/EV-0007 for complete details on the ACB). The Dataway Access Port allows a system controller, via the CAMAC Command Highway, to read and write ACC memory during program load or execution times.

Several different memory configurations are possible, depending upon the nature of the ACC application. One may operate only with the RAM (1K from 0-3776$_8$) and PROM (1K from 170000$_8$ to 173776$_8$) that reside on the LSI-11/2 processor board, or additional memory may be plugged in. A fast (50-ns) 1K board, and a medium speed (200-ns) 4 or 8K board is provided to accommodate the data-acquisition users, while the slower 16K DEC dynamic memory board and a 4 or 8K PROM memory board are available for control applications. A tape cassette unit and control panel have been designed for control applications, and will be described in a future paper dealing with stand-alone beam line control using the ACC.

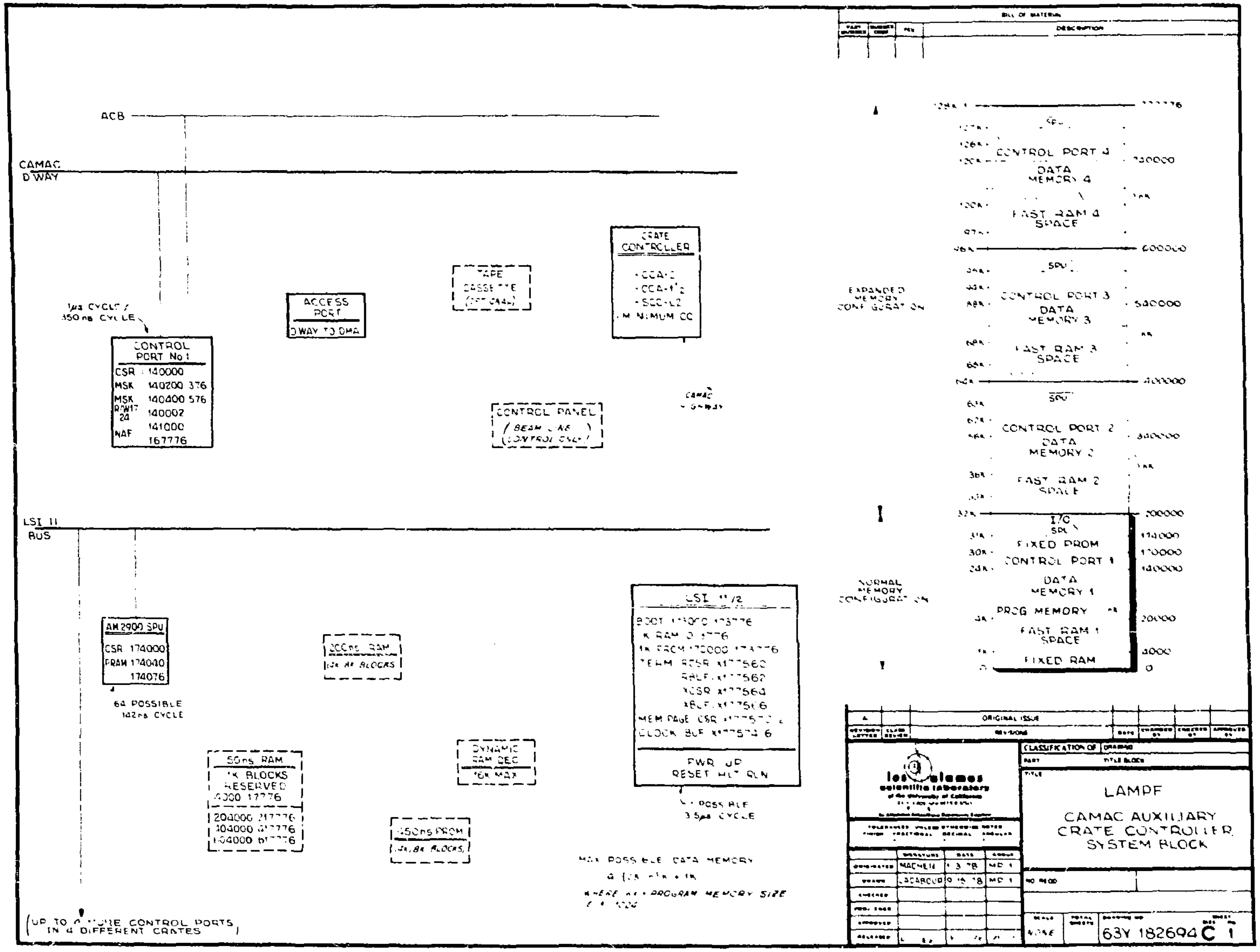


Fig. 2. Block diagram of ACC and address space allocation (63Y-182694/1).

Figure 2 also shows the address space allocation for the LAMPF ACC. Logic has been included in the processor module to easily switch the address space to one of four 32K segments. The purpose of this is not only to allow for a great deal of memory, but to easily accommodate multiple Control Ports and Special Processor Units (SPUs). The address space allocation indicates how fast ram and data memory could be spotted throughout the available 128K address space. Further, multiple Control Ports and multiple SPUs can be assigned address space in each 32K address page.

Figure 3 indicates how, physically, the ACC extends to more than one CAMAC crate when an expanded dataway is required for some specific system problem.

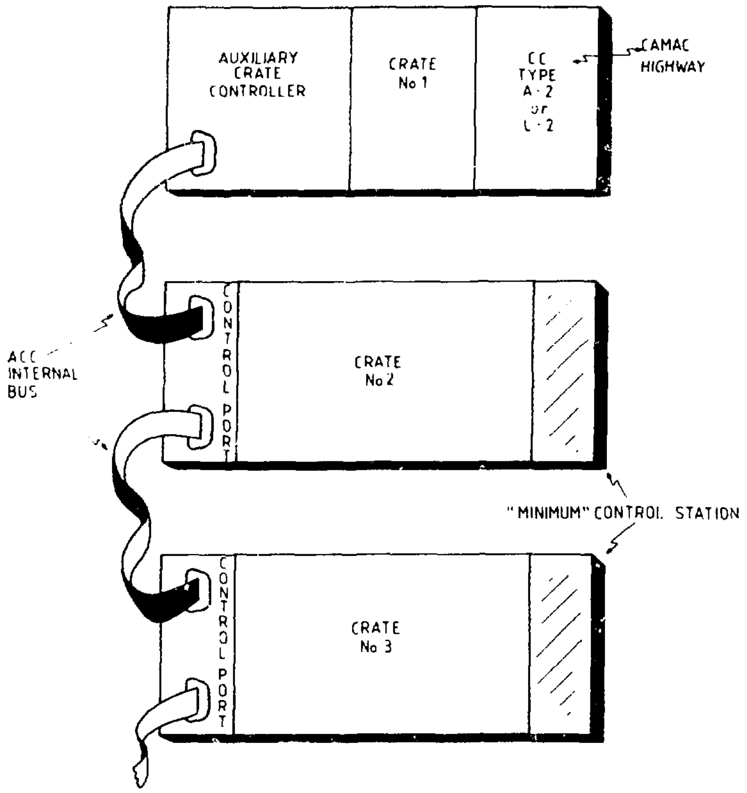


Fig. 3. Multicrate extension.

Note that the single-width Control Port interacts with a "minimal" controller only. The minimal controller consists of an N-Decode and L-patch in the simplest case, and an N-Decoder plus L-grader in the more elaborate case. A description of the minimal controller is included at the end of Section II of this report.

Finally, a comment should be made concerning the selection of the LSI-11/2 as the primary processing elements for the ACC. One could argue that nearly any single-chip microprocessor could be incorporated successfully into the ACC; however we have found that a 16-bit word length machine has a significant advantage in speed and ease of programming over the various 8-bit devices on the market today. Further, LASL in general and LAMPF specifically uses a great number of the PDP-11 family computers. The software correspondence cannot be overlooked and was a major factor in selecting the LSI-11/2 for use in the ACC.

II. PROTOTYPE ACC BOARDS

Each device noted in Fig. 2 making up the ACC will be described from its physical design standpoint in addition to its operational aspects. In the case of the AMD-2900 Special Processor Unit, the microinstruction set will also be covered.

A. LSI-11/2 Processor Board

This two-wide module is the heart of the ACC and contains sufficient hardware to operate in a stand-alone mode should a computing facility of limited capability be desired.

As viewed from the front panel, the LSI-11/2 resides on the left side, while the remainder of the board logic is on the right board as is the dual 43-pin card edge connector providing access to the Q-Bus. This connector is identical to the Dataway connector located in the rear of the card, but raised a few centimeters vertically from the Dataway connector. The power requirements are:

	+5 V	+12 V
Left side (LSI-11/2)	1.2 A	0.22 A
Right side	1.1 A	

(This board also uses -12 V and +24 V.)

The board contains the following manual controls:

RESET pushbutton	- Causes the LSI-11/2 tc execute a Q-Bus $\overline{\text{INIT}}$, and force a program jump to 173000_8.
HALT/RUN	- When this switch is in RUN, the 11/2 is <u>enabled</u> to RUN only. When the switch is in the HALT position, control will be transferred to microcode ODT. A terminal device must be connected to interact with ODT. See DEC manual for operating details.
BOOT MODE	- This switch controls the state of a status register bit to allow software control of the boot-up program source. See <u>terminal</u> details.

Q-BUS CONNECTIONS TO ACC FRONT CONNECTOR

DEC Pin No.	Signal	ACC Connector No.
AU2	BDAL 0L	1
AV2	" 1L	2
BE2	" 2L	3
BF2	" 3L	4
BH2	" 4L	5
BJ2	" 5L	6
BK2	" 6L	7

Q-BUS CONNECTIONS TO ACC FRONT CONNECTOR
(continued)

DEC Pin No.		Signal		ACC Connector No.	
BL2		BDAL 7L		8	
BM2		" 8L		9	
BN2		" 9L		10	
BP2		" 10L		11	
BR2		" 11L		12	
BS2		" 12L		13	
BT2		" 13L		14	
BU2		" 14L		15	
BV2		" 15L		16	
DEC Memory	AC1 }	" 16L }	extended	17 }	on ACC
only	AD1 }	" 17L }	address	18 }	boards
AK2		BWTBTL		19	
-		SL1		20	
-		SL2		21	
-		SL3	special	22	
-		SL4	LAM	23	
-		SL5	lines*	24	
-		SL6		25	
-		+6 V, 2 A		26	*
AJ2		BSYNCL		27	
AH2		BDINL		28	
AE2		BDOUTL		29	
AF2		BRPLYL		30	
AN1		BDMRL		31	
-		BDMGIL		32	**
AS2		BDMGOL		33	
AL2		BIRQL		34	
-		BIAKIL		35	**
AN2		BIAKOL		36	
AP2		BBS7L		37	
BN1		BSACKL		38	
BR1		BEVNTL		39	
BA1		BDCOKH		40	
AP1		BHALTL		42	
AT2		BINITL		43	
AH1		RUNL		-	

*Source from control port
**Boards not using DMA or INTERRUPT must jumper 32-33 and 35-36, respectively.

Q-BUS CONNECTIONS TO ACC FRONT CONNECTOR (continued)

DEC Pin No.	Signal	ACC Connector No.
BM1 AJ1 AL1 BT1 AM1 AC2 AT1 BC2 BJ1	GROUND	101 - 143
BV1 BA2 AA2	+5 V	-
AD2 BD2	+12 V	-

B. LSI-11/2 Board Hardware

1. Fixed PROM and Boot Memory. A 1K UV-erasable, Programmable Read Only Memory (PROM) is available at address:

170000_8 to 173776_8 (30 - 31K) .

The bootstrap code chosen for a specific application must begin at 173000_8 and may extend to the top of PROM. Again, the LSI-11/2 is set to transfer control to this boot location on RESET or power-up. The remaining free locations in on-board PROM may be used at the discretion of the system designer. Further, this PROM is only available on address page 0; the equivalent space on address page 1, 2, or 3 cannot be assigned.

2. Fixed RAM. A 1K Random Access Memory (RAM) is available at address:

0 to 3776_8 (0 - 1K) .

NOTE: This memory may be disabled by switch option on the pc board.

As with the fixed PROM, this RAM is only available on address page 0 and the equivalent space on page 1, 2, or 3 cannot be assigned. Memory was included in the low end of address space in order that any given system will always have trap, interrupt, and stack space when needed. In addition, a certain amount of program space is available and may be used for small problems.

3. Memory Page Control. Control of the memory paging logic is available at Control/Status Register (CSR) address:

$X1177570_8$,

where X1 indicates that the address responds for memory page control on any page (i.e., X1 = 1, 3, 5, 7).

	7	6	5	4	3	2	1	0		
PAGE CONTROL CSR	-	CIE	-	-	-	-	PE1	PE0	READ OR WRITE	X177570$_8$

PE1	PE0	Base Address	
0	0	Page 0	0
0	1	Page 1	200000$_8$
1	0	Page 2	400000$_8$
1	1	Page 3	600000$_8$

In order to conserve space on the processor module, the interrupt enable bit for the multifrequency clock was included in the CSR for the memory page logic. The reader is referred to the next section for the meaning of CIE. The base address is cleared to 0 on power-up or RESET.

4. Multifrequency Clock. A free-running, 16-bit multifrequency clock is available to be read at address:

X177574$_8$.

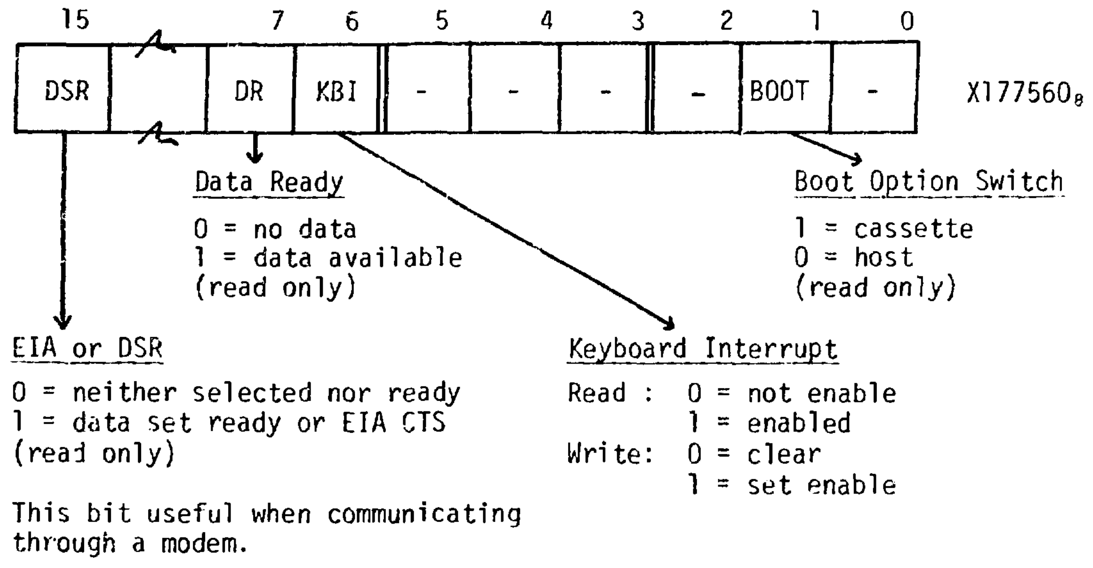
Again, X indicates that the clock may be accessed at any page address. The clock frequency can be switch-selectable at 100 Hz or 1000 Hz. The frequencies are generated by adjustable free-running oscillators (VCOs) on the processor board. The clock is cleared to 0 on power-up or RESET.

Clock Interrupt--the (100-Hz or 1000-Hz) clock input can be routed to the LSI-11/2 EVENT Interrupt by setting CIE (bit 6) at X177570$_8$ to logic 1. Provided PS bit 7 is clear (logic 0) and an appropriate vector is contained at address 100$_8$ and 102$_8$, the LSI-11/2 will interrupt to that vector on every clock tick. Alternatively, the clock may be read when desired to provide a measure of elapsed time.

5. Terminal Port. A multidevice terminal part is available on a rear 36-pin card edge connector. The connector is physically on the back of the rightmost pc card when viewing the module from the front of the CAMAC crate.

The terminal port is configured to operate EIA standard terminals, a 20-mA teletype/paper tape, or devices using TTL levels. The port control is cleared on power-up or RESET.

- An 8-bit character may be read at address X177562$_8$.
- An 8-bit character may be transmitted at address X177566$_8$.
- A Receive CSR is available at address X177560$_8$.



NOTE: Unused bits are read as a logic 1.

A Transmit CSR is available at address X177564₈ .

Read only - bit 7 is Transmit buffer empty.

0 = full
1 = empty

NOTE: Unused bits are read as a logic 1.

Keyboard Interrupt

When enabled (bit 6 of RCSR = 1), the LSI-11/2 may receive an interrupt to:

Vector at 60_8, 62_8 ,

provided the PS bit 7 = 0 and the vector at 60_8 to 62_8 contains the proper information.

Baud Rates

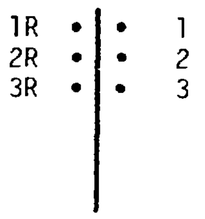
The terminal port may be set to transmit and receive at a number of different baud rates set by switch located on the processor pc board.

Switch	4	3	2	1	Baud Rate
	0	0	1	0	110
	0	1	0	1	300
	0	1	1	1	1200
	1	0	1	0	2400
	1	1	0	0	4800
	1	1	1	0	9600

0 = a closed switch

Terminal Port Connector

The rear-mounted 36-pin edge connector is wired in the following way:



Viewed from rear of module

Pin	Signal	Pin	Signal
1	20 mA Out +	1R	20 mA Out -
2	20 mA In -	2R	Ground
3	TTL Out	3R	"
4	TTL In	4R	"
5	EIA Out	5R	"
6	EIA In	6R	"
7	EIA RTS/DTR	7R	"
8	EIA CTS	8R	"
9	EIA DSR	9R	"
10	Reader Start +	10R	Reader Start -

The terminal port provides the following ASCII character handling:

1. Parity is neither generated nor checked.
2. Two stop bits are generated and checked.
3. Eight data bits are sent and received.
4. HALTL is asserted on a Frame Error.

C. Dataway Access Port

When an ACC is equipped with a Dataway Access Port, the system controller, via the Crate Controller and CAMAC Command Highway (see Fig. 1), is able to access memory and control the operation of the ACC. A Memory Address Register (MAR) and Memory Data Register (MDR) hold ACC bus cycle information during a Direct Memory Access (DMA) of the Q-bus. Owing to coupling latencies between the Dataway, the Q-bus, and the CAMAC Highway/System Controller, a requested Read or Write cycle is carried out when the Dataway cycle completes. For this reason, Read data must be acquired on the next CAMAC cycle addressed to the Access Port.

CAMAC Commands

The Access Port responds to a set of NAFs (listed below) and will return X = 1 to the system controller provided a command within the valid set is received.

Subaddress A0

DIS (F24) - Disable RUN and enter ODT.

XEQ (F25) - Bus INIT and Enable RUN.

The following notes pertain to the use of XEQ·A0:

	Condition	Action	Result
1.	LSI-11/2 not in ODT	Issue XEQ·A0	Bus Init Generated by Access Port (no action to 11/2, only others)

	Condition	Action	Result
2.	LSI-11/2 is in ODT	Issue XEQ·A0	Enable RUN only 11/2 stays in ODT (see XEQ·A1 to boot)

Subaddress A1

XEQ - Initiates RESET to 11/2 and forces program jump to 173000_8 boot. 11/2 issues BUS INIT.

Subaddress A2

RD2 (F1) - Read MAR (R16-R1)

WT2 (F17) - Write MAR (W16-W1) and initiate DMA READ cycle at MAR.

Q = 1 ACC DMA cycle able to proceed.

Q = 0 reissue CAMAC command as the DMA cycle was not able to proceed due to other ACC bus activity.

Subaddress A3

RD1 (F0) - Read MDR (R16-R1)

This command must follow a successful TST·A4 to ensure that a previous read request took place.

WT1 (F16) - Write MDR (W16-W1) and initiate a DMA WRITE cycle to the ACC bus.

Q = 1 MDR write took place and DMA cycle able to proceed. Following the DMA cycle MAR ← MAR+2.

This command must be followed by a TST·A4 to determine when the DMA cycle is complete.

XEQ - MAR ← MAR+2 then initiates a DMA READ cycle to the ACC bus.

Q = 1 increment took place and the DMA cycle is able to proceed.

This command must also be followed by a TST·A4 to determine when the DMA cycle is complete.

Subaddress A4

TST (F27) - Q = 1 indicates that the requested cycle took place and that no other device request is pending.

Subaddress A12

RD2 - Read the ACC status register (R3-R1)

R1 = 1 Dataway Access Port cycle in progress.

R2 = 1 Another ACC device is bus master.

R3 = 1 HALT asserted (either by Access Port logic or by switch on LSI-11/2 module).

Dataway C·S2 generates an internal Access Port reset and Z is not responded to.

Operating Philosophy

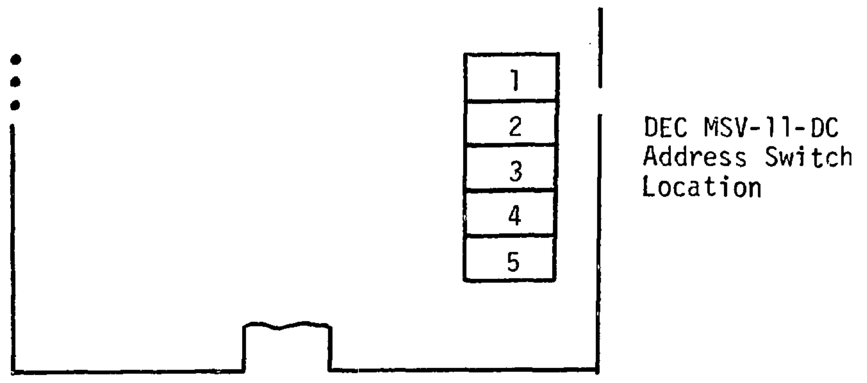
Since Read and Write operations are accomplished by DMA bus sharing, it is quite possible for the System Controller to read-out or load a section of ACC Memory while other operations are under way. I would caution the user, however, to halt ACC activity if a great deal of interrupt or other DMA activity is going on. This activity will only show the Access Port response and create a latency that may not be acceptable. Provided the user knows what the ACC is up to, shared Access Port operation is permissible.

I also caution the user in the careful use of the various Q tests and the appropriate application of the TST·A4 command. Always check Q and issue the TST command at the required intervals when executing Read and Write ACC bus cycles.

D. ACC Memory

The various memory options available have been discussed in part in Section I of this report. In this section, the address compare switch settings will be presented, along with some comments concerning the types of memory and their ultimate use in ACC systems.

One additional memory should be mentioned as it will find use in certain control applications of the ACC. This is the DEC MSV-11-DC 16K MOS Dynamic RAM. The memory is packaged on a 1-wide CAMAC module with a single DEC connector inside the module for the memory board, and a 43-pin ACC bus connector available at the front of the module. The unit draws 1.7 A at +5 V and 0.34 A at +12 V. Five switches are located on a "DIP Switch" near the board connector, as shown below.



A "closed" switch = 0 or on, and an "open" switch = 1 or off.

	Switch No.				
Starting Address$_8$	1	2	3	4	5
20000	0	0	0	0	1
40000	0	0	0	1	0
60000	0	0	0	1	1
100000	0	0	1	0	0
120000	0	0	1	0	1
140000	0	0	1	1	0
160000	0	0	1	1	1
200000	0	1	0	0	0
220000	0	1	0	0	1
240000	0	1	0	1	0
260000	0	1	0	1	1
300000	0	1	1	0	0
320000	0	1	1	0	1
340000	0	1	1	1	0
360000	0	1	1	1	1
400000	1	0	0	0	0
420000	1	0	0	0	1
440000	1	0	0	1	0
460000	1	0	0	1	1
500000	1	0	1	0	0
520000	1	0	1	0	1
540000	1	0	1	1	0
560000	1	0	1	1	1
600000	1	1	0	0	0
620000	1	1	0	0	1
640000	1	1	0	1	0
660000	1	1	0	1	1
700000	1	1	1	0	0

This memory must not be started at 0, as RAM already exists from 0 to 3776$_8$ on the LSI-11/2 board, unless that RAM has been disabled by switch option.

200-ns RAM

In applications where a faster memory cycle time is required and/or the random latencies generated by the dynamic RAM refresh operation cannot be tolerated, a 4 to 8K (depending upon the number of memory chips plugged in) static RAM is available. The memory contains AMD-9130 1024 x 4 static RAM chips (16 for 4K or 32 for 8K) and can be loaded with any multiple of 1K words as long as the system application can accept a 4K or 8K block address space assignment independently of the actual amount of RAM implemented.

An 8-switch "DIP-switch" is located on the module from which six switches are used to select the starting address and the 4K or 8K option. Again, a closed switch, or on, = logic 0 and an open switch, or off, = logic 1. Switch 6 controls the 4K to 8K option. The unit is a 1-wide module, drawing 1.8 A at +5 V for the 4K option.

4K Option, Switch 6 = 1

Start Address	Switch No. 1	2	3	4	5
20000	0	0	0	0	1
40000	0	0	0	1	0
60000	0	0	0	1	1
100000	0	0	1	0	0
120000	0	0	1	0	1
140000	0	0	1	1	0
160000	0	0	1	1	1
200000	0	1	0	0	0
220000	0	1	0	0	1
240000	0	1	0	1	0
260000	0	1	0	1	1
300000	0	1	1	0	0
320000	0	1	1	0	1
340000	0	1	1	1	0
360000	0	1	1	1	1
400000	1	0	0	0	0
420000	1	0	0	0	1
440000	1	0	0	1	0
460000	1	0	0	1	1
500000	1	0	1	0	0
520000	1	0	1	0	1
540000	1	0	1	1	0
560000	1	0	1	1	1
600000	1	1	0	0	0
620000	1	1	0	0	1
640000	1	1	0	1	0
660000	1	1	0	1	1
700000	1	1	1	0	0
720000	1	1	1	0	1
740000	1	1	1	1	0
760000	1	1	1	1	1

8K Option, Switch 6 = 0

Start Address	Switch No. 1	2	3	4	5
40000	0	0	0	1	X
100000	0	0	1	0	X
140000	0	0	1	1	X
200000	0	1	0	0	X
240000	0	1	0	1	X
300000	0	1	1	0	X
340000	0	1	1	1	X
400000	1	0	0	0	X
440000	1	0	0	1	X
500000	1	0	1	0	X
540000	1	0	1	1	X
600000	1	1	0	0	X
640000	1	1	0	1	X
700000	1	1	1	0	X
740000	1	1	1	1	X

X = Doesn't care

50-ns RAM

A 1K 50-ns fast RAM has been designed using a 1K x 1 Fairchild 93L425 static RAM chip. This memory will find use in ACC front-end data acquisition and filtering applications where fast buffer store is required.

The unit is a 1-wide CAMAC module requiring 1.2 A at +5 V.

An 8-switch "DIP-switch" is located on the module from which five switches are used to select the starting address, which can be spotted in either low or high memory in any of the four address pages. A closed switch, or on, = logic 0, and an open switch, or off, = logic 1. Switches 1 and 2 control the page boundary, switch 3 controls low or high memory placement, and switches 4 and 5 control the 1K boundary. Again, this memory should not be placed at address 0, as memory at that address already exists on the LSI-11/2 board, unless the address 0 RAM has been disabled by switch option.

	Start Address$_8$		Switch No.				
			1	2	3	4	5
PAGE 0	4000	lo	0	0	0	0	1
	10000		0	0	0	1	0
	14000		0	0	0	1	1
	100000	hi	0	0	1	0	0
	104000		0	0	1	0	1
	110000		0	0	1	1	0
	114000		0	0	1	1	1
PAGE 1	200000	lo	0	1	0	0	0
	204000		0	1	0	0	1
	210000		0	1	0	1	0
	214000		0	1	0	1	1
	300000	hi	0	1	1	0	0
	304000		0	1	1	0	1
	310000		0	1	1	1	0
	314000		0	1	1	1	1
PAGE 2	400000	lo	1	0	0	0	0
	404000		1	0	0	0	1
	410000		1	0	0	1	0
	414000		1	0	0	1	1
	500000	hi	1	0	1	0	0
	504000		1	0	1	0	1
	510000		1	0	1	1	0
	514000		1	0	1	1	1
PAGE 3	600000	lo	1	1	0	0	0
	604000		1	1	0	0	1
	610000		1	1	0	1	0
	614000		1	1	0	1	1
	700000	hi	1	1	1	0	0
	704000		1	1	1	0	1
	710000		1	1	1	1	0
	714000		1	1	1	1	1

The reader is referred to Fig. 2, address space allocation, to see graphically where the 1K fast RAM may be placed.

450-ns PROM

Applications involving the ACC in stand-alone or partially stand-alone control systems will dictate that checked-out code be placed in nonvolatile read only memory. In order that a small quantity of this memory be economically feasible and practical for laboratory use, an ultra violet erasable PROM (Intel 2716, 2K x 8) was selected. The cycle time is 450 ns and the 4K/8K memory is constructed on a 1-wide CAMAC module, drawing 1 A at +5 V. Programming of the PROM chip is accomplished on any of the microprocessor development machines available at LAMPF.

An 8-switch "DIP-Switch" is located on the module from which six switches are used to select the starting address and the 4K or 8K option. The closed, or on, switch = logic 0, and the open, or off, switch = logic 1. Switch 6 controls the 4K/8K option.

4K Option, Switch 6 = 1

Start Address	Switch No. 1	2	3	4	5
20000	0	0	0	0	1
40000	0	0	0	1	0
60000	0	0	0	1	1
100000	0	0	1	0	0
120000	0	0	1	0	1
140000	0	0	1	1	0
160000	0	0	1	1	1
200000	0	1	0	0	0
220000	0	1	0	0	1
240000	0	1	0	1	0
260000	0	1	0	1	1
300000	0	1	1	0	0
320000	0	1	1	0	1
340000	0	1	1	1	0
360000	0	1	1	1	1
400000	1	0	0	0	0
420000	1	0	0	0	1
440000	1	0	0	1	0
460000	1	0	0	1	1
500000	1	0	1	0	0
520000	1	0	1	0	1
540000	1	0	1	1	0
560000	1	0	1	1	1
600000	1	1	0	0	0
620000	1	1	0	0	1
640000	1	1	0	1	0
660000	1	1	0	1	1
700000	1	1	1	0	0
720000	1	1	1	0	1
740000	1	1	1	1	0
760000	1	1	1	1	1

8K Option, Switch 6 = 0

Start Address	Switch No. 1	2	3	4	5
40000	0	0	0	1	X
100000	0	0	1	0	X
140000	0	0	1	1	X
200000	0	1	0	0	X
240000	0	1	0	1	X
300000	0	1	1	0	X
340000	0	1	1	1	X
400000	1	0	0	0	X
440000	1	0	0	1	X
500000	1	0	1	0	X
540000	1	0	1	1	X
600000	1	1	0	0	X
640000	1	1	0	1	X
700000	1	1	1	0	X
740000	1	1	1	1	X

X = Doesn't Care

E. Control Port

The ACC Control Port (CP) performs CAMAC Dataway command operations in response to addressed DATI (Read) and DATO (Write) Q-bus cycles. A Status Register controls the mode of operation with respect to Dataway cycles and LAM response. A normal LAM source has been provided which can be enabled from the Dataway (by the System Controller) and set by the ACC. This feature allows an ACC to gain the attention of the system controller in an assynchronous way for the purpose of passing information or warning messages, etc.

The user must be familiar with the connection options for priority arbitration of Dataway Mastership. The LAMPF ACC follows the requirements of DOE/EV-0007 for generation and reception of Request/Grant signals and reception of the Auxiliary Controller Lockout signal.

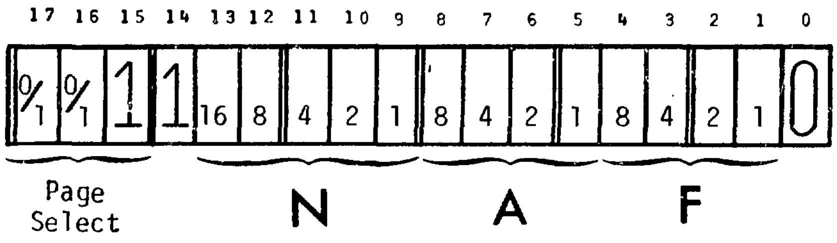
A 4-switch "DIP-Switch" is located on the module from which two switches are used to select the page boundary address of the CP. The closed, or on, switch = logic 0, and the open, or off, switch = logic 1.

CP Address Range	Switch No. 1	Switch No. 2
140000 - 167776	0	0
340000 - 367776	0	1
540000 - 567776	1	0
740000 - 767776	1	1

Using the boundary setting switches, one can place up to four control ports with one ACC, one CP per CAMAC crate, and effectively extend the primary crate space.

The Control Port is constructed on a 1-wide CAMAC module and requires 1.48 A at +5 V.

CAMAC NAF



The address allocation for a Mapped CAMAC NAF, with NØ and N24-N31 being illegal addresses within a CAMAC crate, is:

X141000$_8$ - X167776$_8$.

The mapped address to NAF table then looks like:

N1A0F0 → X141000

N1A0F1 → X141002

⋮ ⋮

N23A15F15 → X167776 .

The following Q-bus cycles, along with F(8) bit, provide for the switching between the various types of CAMAC functions:

Q-Bus Cycle	F(8)	Dataway Operation
DATI (READ)	0	F0-F7 READ
"	1	F8-F15 CONTROL
DATO (WRITE)	0	F16-F23 WRITE
"	1	F24-F31 CONTROL

DATI bus cycles to the CP (F0-F7) access the CAMAC R lines R16-R1, and latch R24-R17 into a holding register for later access. DATI bus cycles to the CP (F8-F15) transfer Q on bit 0 and X on bit 1 for the Dataway cycle executed.

DATO bus cycles to the CP (F16-F23) will transfer bus data to W16-W1 and the contents of the W24-W17 holding latch to W24-W17. A DATO bus cycle to the CP (F24-F31) will <u>only</u> execute the Dataway cycle, and Q/X for that operation must then be read from the CP Status Register.

<u>R/W 17-24 Holding Register</u>
The high-order 8 bits associated with the Read and Write Dataway Operation are accessed as follows:

<u>Q-BUS CYCLE</u>	7	6	5	4	3	2	1	0	<u>ADDRESS</u>
DATI (READ)	R24	R23	R22	R21	R20	R19	R18	R17	X140002$_8$

The R24-R17 lines are held during a READ CAMAC operation and may be read-out after the cycle is complete.

	7	6	5	4	3	2	1	0	
DATO (WRITE)	W24	W23	W22	W21	W20	W19	W18	W17	X140002$_8$

The W24-W17 buffer must be written prior to the WRITE CAMAC Operation, if these lines are used in the transacti

```
Example:  MOV @#141000,R0   ;   N1A0F0   lo byte
          MOV @#140002,R1   ;   N1A0F0   hi byte

          and

          MOV R1,@#140002   ;   WRITE    hi byte
          MOV R0,@#141000   ;   N1A0F16
```

<u>The CP Status Register</u>
The Status Register can be accessed at address:

X140000$_8$,

and is useful in controlling the operational mode of the CP, in addition to providing access to some special features of the LAMPF ACC.

	8	7	6	5	4	3	2	1	0	
DATI (READ)	EI2	EI1	SC	MC	I*	LE	L	X	Q	X140000$_8$

Bits 9-15 are read as 1s.
Q and X are from the last Dataway cycle executed.

L	is the state of the CP L-Source.
LE	is the state of the CP L-Mask.
I*	is the state of Dataway Inhibit.
MC	is the state of the Multi-cycle mode control.
SC	is the state of the Short-cycle mode control.
EI1	is the state of the 11/2 interrupt logic enable.
EI2	is the state of the SPU LAM bus enable.

	8	7	6	5	4	3	2	1	0	
DATO (WRITE)	EI2	EI1	SC	MC	I	C	Z	-	L	$X140000_8$

Bits 9-15 are not used.

L = 1	sets the CP L-Source to logic 1.
Z = 1	initiates a Dataway Z.
C = 1	initiates a Dataway C.

NOTE: Both Z and C must be executed in normal cycle mode. The CP logic momentarily sets normal cycle during the period of these commands, and returns to short-cycle mode if that mode had been invoked prior to Z or C being given.

I = 1	sets and holds the Dataway Inhibit line.
MC = 1	sets Multi-cycle mode.
SC = 1	sets Short-cycle mode.
EI1 = 1	enables the 11/2 interrupt logic.
EI2 = 1	enables the SPU LAM bus.

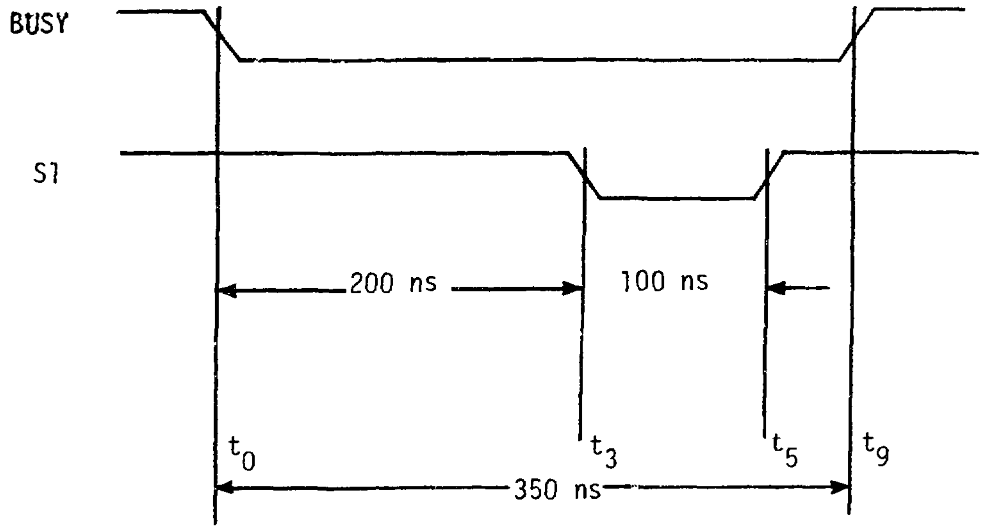
Multi-Cycle Mode

An Auxiliary Controller is allowed to execute multi-cycle control of the Dataway provided all other priority arbitration rules are met (see DOE/EV-0007). In multi-cycle mode, the CP initially arbitrates for Dataway Mastership preceding the first Dataway cycle. Once Dataway Busy is asserted, it remains asserted until Multi-cycle mode is cleared and the control port then executes single-cycle operations. An ACC operating in Multi-cycle mode has fewer overhead latencies per Dataway cycle and is, therefore, somewhat faster than one operating in single-cycle mode.

Short-Cycle Mode

The LAMPF ACC is capable of executing 350-ns Dataway cycles once SC = 1. This cycle does not contain strobe S2, and all other times (see Fig. 9 of IEEE Standard 583-1975) are considerably shortened. The following sketch indicates the Dataway timing for the SC mode.

Clearly, a CAMAC module accessed by the CP in Short-cycle mode must be able to respond in the times indicated above. This mode is useful in high data-rate applications.



LAM Handling
The CP provides two parallel paths for coping with LAMS within the crate:

- One path, enabled by EI1 in the Status Register, priority encodes 23 LAM lines (L1 - highest priority, L23 = lowest priority) and presents the resultant 5-bit code to the LSI-11/2 as an interrupt vector address. Base Address = 400_8.

- The second path, enabled by EI2 in the Status Register, places up to six selected LAMS on a special set of six lines in the expanded ACC Q-bus. These lines can be tested by a Special Processor Unit (SPU) for use in high-speed data system requirements. More will be said about this technique in Section IIF.

A RAM mask provides the CP masking for LSI-11/2 interrupts. A 64 x 1 RAM is available at address:

X140200_8 - X140376_8 .

Clearly, only the first 24 locations are of concern, as the RAM is addressed by the encoded LAM pattern of the crate. However, the other 40 locations can be used as a 1-bit store for special code, as required by the user.

Prior to writing the mask, EI1 must = 1.

- A DATO cycle to X1402XX will write the state of bit 1 into the RAM at location XX. A logic 1 enables the corresponding LAM.

- A DATI cycle to X1402XX will read the state of location XX into bit 1 of the destination.

The following table contains the correspondence between LAM-Mask Address and 11/2 vector for the first LAM handling scheme.

	LAM	Mask Address	Vector Address	
	-	X140200	400-402	Error Vector
Lowest - priority	23	X140202	404-406	
	22	204	410-412	
	21	206	414-416	
	20	210	420-422	
	19	212	424-426	
	18	214	430-432	
	17	216	434-436	
	16	220	440-442	
	15	222	444-446	
	14	224	450-452	
	13	226	454-456	
	12	230	460-462	
	11	232	464-466	
	10	234	470-472	
	9	236	474-476	
	8	240	500-502	
	7	242	504-506	
	6	24[illegible]	510-512	
	5	246	514-516	
	4	250	520-522	
	3	252	524-526	
	2	254	530-532	
Highest - priority	1	256	534-536	

The SPU-LAM Bus

The second LAM Mask RAM is located at address:

X140400 - X140576

and is a 64 x 1 memory that is addressed by six patchable LAMS selected by the user. Further, the user must decide what combinations of the six selected (or less) LAMS will be masked on. If a given combination of LAMS is enabled (that address contains a logic 1), then the six LAMS are placed on six special Q-Bus LAM lines. The lines will be asserted within 100 ns of the assertion of any particular LAM.

Prior to writing the Mask, EI2 must = 1.

- A DATO cycle to X1404XX will write the state of bit 1 into the RAM 2 at location XX. A logic 1 enables the corresponding LAM pattern.
- A DATI cycle to X1404XX will read the state of location XX to bit 1 of the destination.

Switch 4 of the CP Address select "DIP-Switch" is an enable/disable control for the RAM2.

An Open SW4 = Normal operation as above.
A Closed SW4 = All 6-bit patterns enabled.

The CP LAM Source

A System Controller, via a Crate Controller in Station 23-24 (or the CP itself), may set up a LAM from the ACC for use in Demand communications with a System Controller. The LAM Mask is addressed via the Dataway at Subaddress 0 and the station number associated with the CP.

Subaddress A0:	ENB (F26)	Enable the LAM
	DIS (F24)	Disable the LAM
	CLM (F10)	Clear the Source
	TLM (F8)	Test the LAM independent of the Mask state

X is returned for each command above.

The LAM-source is controlled by bit - 0 of the CP Status Register in a DATO cycle. Bit 3 of the CP Status Register during a DATI cycle contains the state of the LAM Mask. Thus, once set up, the ACC can send a Demand to a System Controller.

F. The Special Processor Unit

The challenge of systems development, when the ultimate aim is to produce useful hardware to further a given (larger) task, is to utilize the newly available state-of-the-art technology and not to reinvent a wheel. To this end, the 16-bit LSI-11 microcomputer was chosen as the primary control element in the ACC; software compatibility and a reasonable word size were prime factors in the choice (the Ferrenti F100L was chosen for the Daresbury unit for those same reasons at that laboratory).

From the point of view of program execution times, the LSI-11 is no better than other minicomputers. The question which remained was how to improve the overall execution times of a certain subset of available ACC operations. One answer to this question was to utilize the newly available bit-sliced, bipolar microprocessor technology in a microprogrammed subroutine structure which would be capable of operating at better than 200 ns per instruction cycle time. The AMD-2900 family of LSI chips was selected for this purpose, and a microprogramming structure which allows both register transfer operations and a reasonable level of arithmetic capability was designed. The organization and operation of the Special Processor Unit (SPU), as it is called, will be described in this section.

The SPU Organization

The block diagram of the SPU is contained in Fig. 4 and shows how a microcoded controller, Arithmetic Logic Unit (ALU) and bus interface combine to form a fast processing facility capable of using the Q-Bus and, therefore, all devices in an ACC.

The SPU is a 1-wide CAMAC module with Q-Bus front connector and requires 2.95 A at +5 V.

SPU Operation

It is strongly recommended that the reader review the AMD-2900 family technical literature before attempting to go much further in this section. Although the finite state machine (constructed from LSI circuits available from AMD) is not especially complex from an outside view, the overall complexity is enormous! The machine is essentially the control unit of a full-scale computing facility, with a 16-bit ALU added to allow logical and arithmetic operations to be

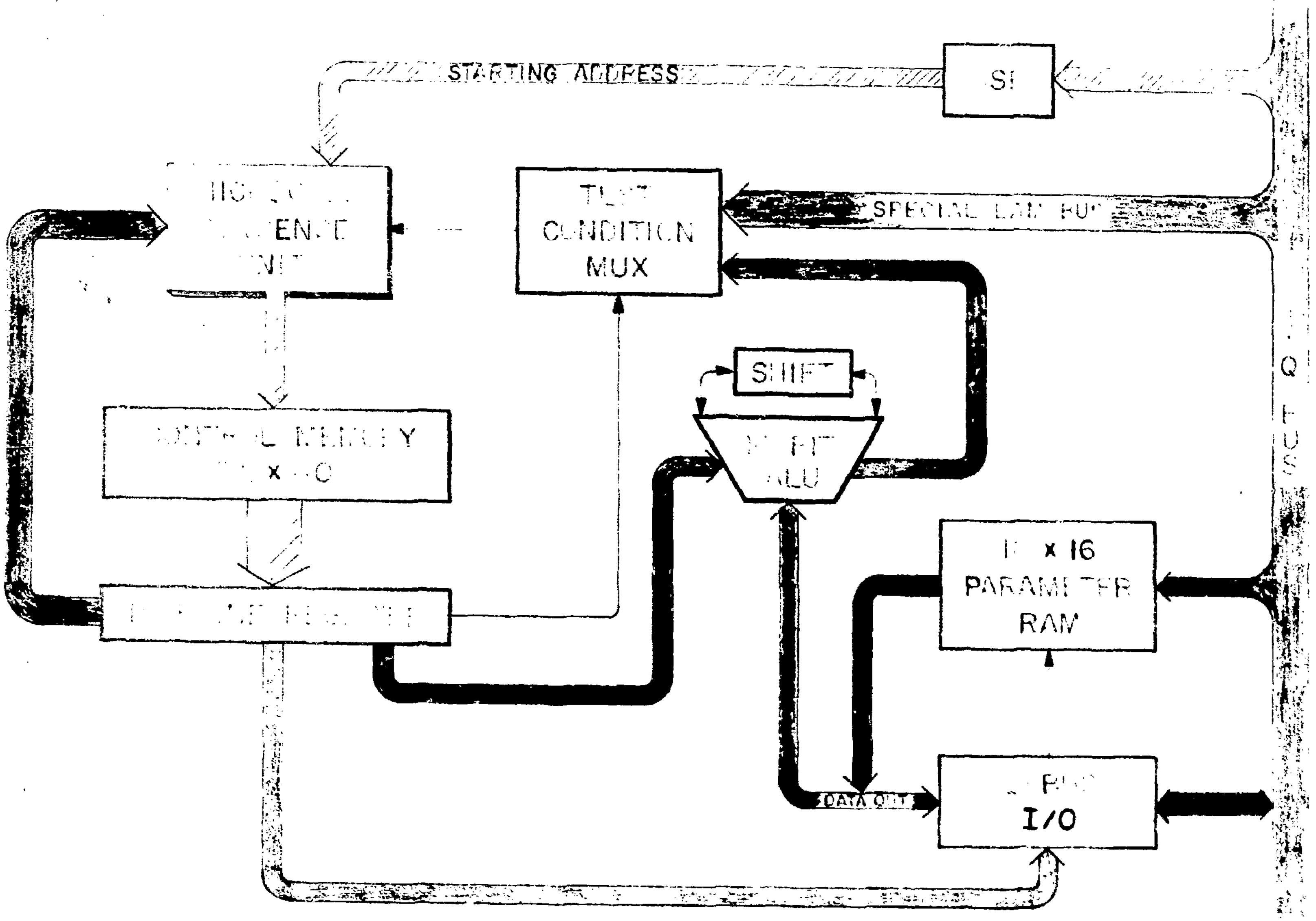


Fig. 4. Special processor organization.

accomplished. The LSI technology which makes such a structure feasible is the 2910 micro-sequencer/address control unit. The 4-bit-wide ALUs (2901A) are designed to operate with the 2910, but are not really special from the point of view of a micro-coded controller. The four ALU slices could just as well have been used with a hardwired control sequencer, and not the firmware controller designed around the 2910.

The finite state machine cycles with a 142-ns period (7-MHz clock) and is synchronous with the rising edge of the clock. The clock duty factor is, by design, 40 to 42% in order to allow more signal set-up time at various chip inputs.

The control memory consists of five 512 x 8 fuse-link PROMs to make up the 512 x 40 memory. Fuse-link PROMs were selected for their fast access time (45 ns), and are considered essential in such a design. The control sequencer is capable of generating a 12-bit "next-address" of which we make use of 9 bits and could expand to 11 bits.

Expansion of the control memory and/or testing of the SPU by PROM simulation has been provided for by two 3M-type connectors on the SPU pc board. A 20-pin connector brings the PROM address plus extended address bits out, and a 40-pin connector brings the 40 micro-instruction bits in. See section IIG for a description of the simulator designed to work into these connectors, in addition to a memory expansion option.

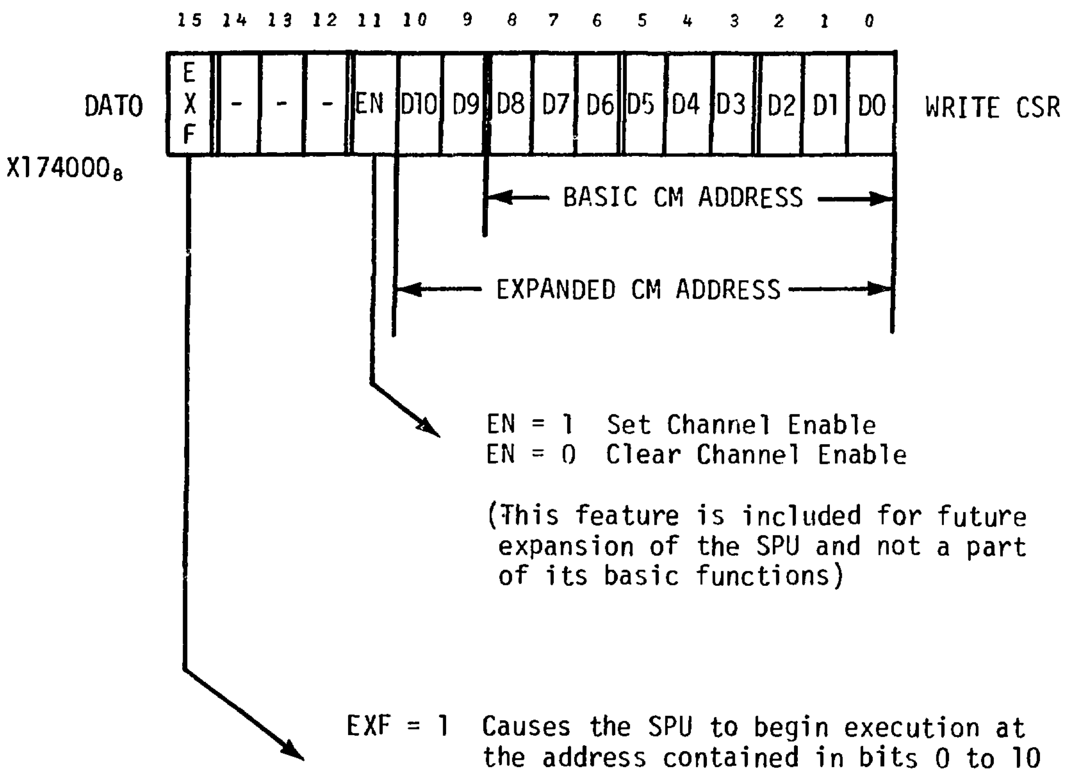
The contents of the control memory (the micro-instructions) are applied to a 40-bit "pipeline" register (PLR) which is clocked on the rising edge of the 7-MHz clock. All machine control is derived from various signals at the output of the PLR. The theory behind a PLR in a finite state machine design is to allow the sequence unit and control memory to settle to the next address during the execution at the current address, thus speeding the instruction time by that settling and access time.

One field of the micro-instruction is a function control to the sequence unit. The instructions will be discussed in Section IIIB, but it is sufficient to say, at this point, that the generation of the next instruction address is controlled through this function code. The next address may be conditional upon some number of test conditions; a field in the micro-instruction word selects these conditions. Finally, the state machine must be told what state to begin execution in; the 11-bit direct input from a Control/Status Register (accessed via the Q-Bus) provides this information.

The CSR is available at address:

X174000$_8$,

and the functions available are:

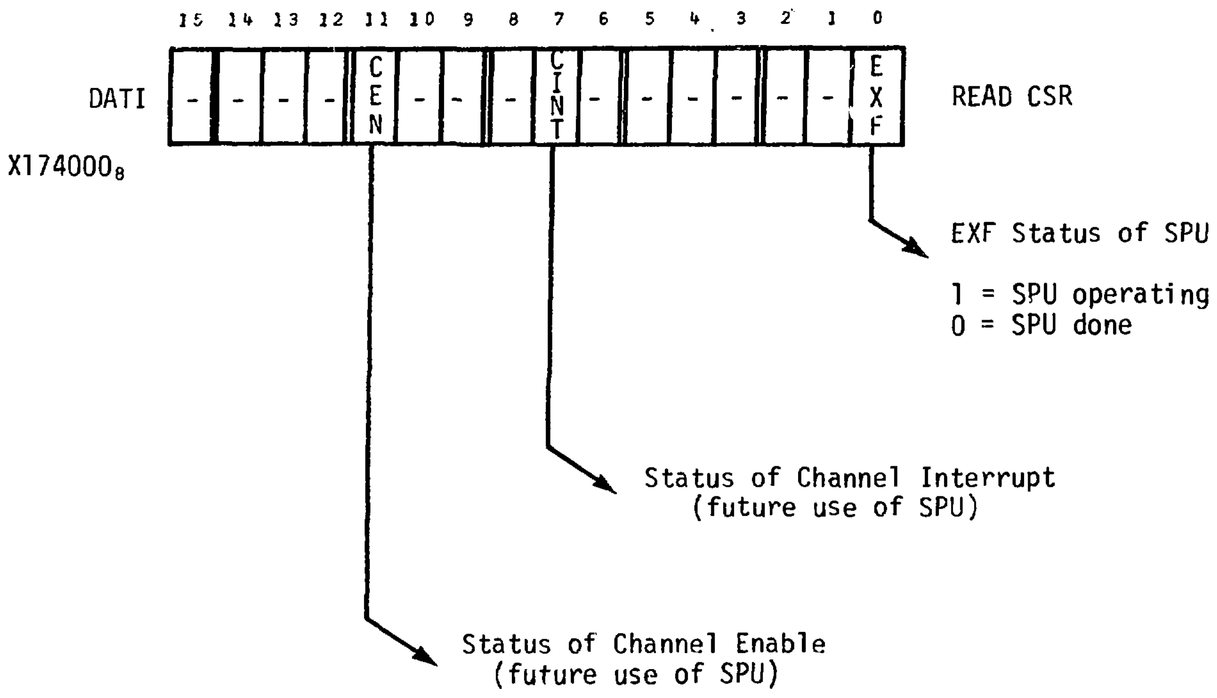


EN = 1 Set Channel Enable
EN = 0 Clear Channel Enable

(This feature is included for future expansion of the SPU and not a part of its basic functions)

EXF = 1 Causes the SPU to begin execution at the address contained in bits 0 to 10 at the CSR word.

EXF = 0 CSR latch only loaded (used for testing and future operation)



A 16-bit ALU with shift function control, look-ahead carry, and 16 input/output signal lines is controlled from another field of the micro-instruction. The ALU function, source and destination (one of 16 on-chip registers or the I/O pins), and direction of data transfer are controlled from this field.

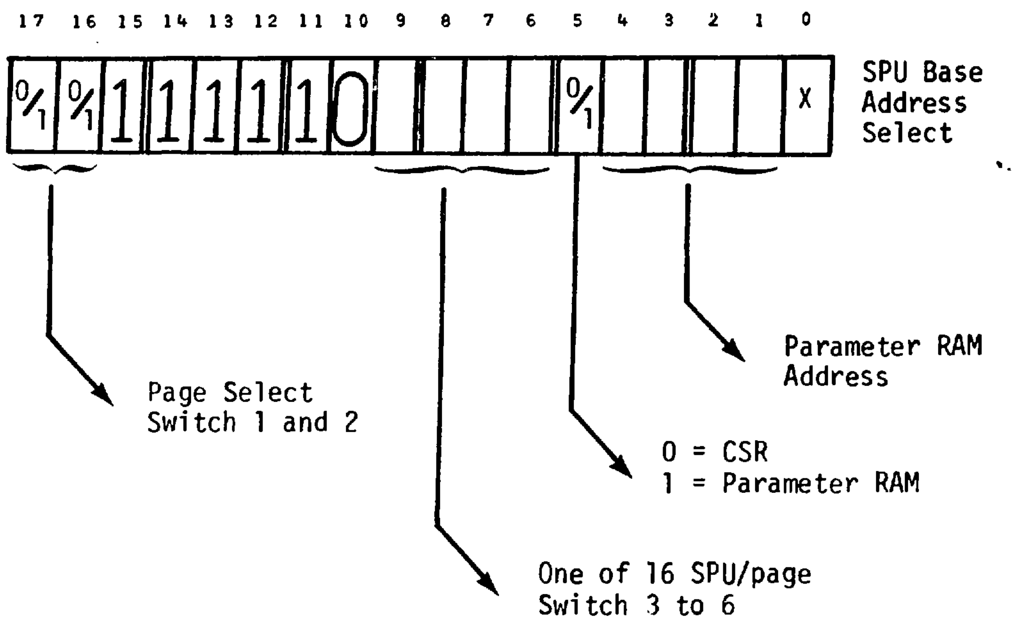
Test conditions from the ALU (zero, negative, etc.) along with six special LAM signals are sent to the test condition Multiplexer previously mentioned. At each instruction time, the sequencer can test a condition that is staticized during the last cycle.

Yet another field of the micro-instruction controls the Q-Bus interface through a control function decoder. The state machine is able to request a DMA hold on the Q-Bus, execute a READ (DATI) or WRITE (DATO) cycle on the bus and transfer data to and from the interface to the ALU. Another important feature of the bus interface is the one-way parameter transfer RAM. As in a software subroutine, we must be able to transfer operating parameters to the firmware subroutine facility. This is accomplished by a 16 x 16 RAM which may be written from the Q-Bus and read by the SPU. The on-chip ALU registers are a reflection of this RAM, and can be used to hold similar working parameters. Note that the RAM cannot be written from the ALU, nor can it be read by the Q-Bus.

The Parameter RAM is available at address:

X174040 - X174076.

An 8-switch "DIP-Switch" located on the module allows the address expansion (page) bits to be set and, in addition, 16 other SPU base addresses may be selected so that a maximum of 16 x 4, or 64, individual SPUs can be operational in one LSI-11/2 controlled ACC.



SPU Base Address		Switch No. 1	2	3	4	5	6
174000		0	0				
374000		0	1				
574000		1	0				
774000		1	1				
	X174000			0	0	0	0
	X174100			0	0	0	1
	X174200			0	0	1	0
	X174300			0	0	1	1
	X174400			0	1	0	0
	X174500			0	1	0	1
	X174600			0	1	1	0
	X174700			0	1	1	1
	X175000			1	0	0	0
	X175100			1	0	0	1
	X175200			1	0	1	0
	X175300			1	0	1	1
	X175400			1	1	0	0
	X175500			1	1	0	1
	X175600			1	1	1	0
	X175700			1	1	1	1

A closed, or on switch = logic 0; an open, or off switch = logic 1.

SPU Timing Measurements

SPU-generated Q-Bus cycle timing measurements are included in the following table. The time recorded is both the SYNC assertion time on the Q-Bus and the number of SPU cycles; a measure of SPU uncertainty is important in measuring the actual speed of operation of a fast register transfer device. Also, the R/W times were measured by asking the SPU to execute a R/W operation a large number of times and then averaging the total time over the number executed.

Source and Destination	Read	Write	R/W
50 ns RAM	225/6	225/6	1250/13
200 ns RAM	575/9	575/9	2000/19
CAMAC (normal cycle)	1000/11	1000/11	2600/23
CAMAC (short cycle)	600/9	600/9	2000/19

The first figure is the actual cycle time in ns and the second is the number of SPU cycles required to complete the cycle.

Future Additions to the SPU

The SPU has been designed and constructed such that options can be added should the need exist. Mention has been made of the channel controller, a 2-MHz bit serial communications-channel controller which formats information into the SDLC or Bi-Sync line protocol. A device of this nature has been foreseen but never actually implemented in an ACC system. Hooks have been left, however, should this form of communication from a remote CAMAC crate be desired (see Ref. 8). Secondly, the necessary connection points have been designated and brought to wirewrap pins such that a 100-ns 8 x 8 combinatorial multiplication chip could be added. This multiplier can greatly speed up the process of multiplication in the SPU, and should be immediately added when it becomes necessary to multiply in a front-end preprocessing application.

Finally, by use of the PROM simulator connectors, an expanded control memory (up to 2K words) can be added to the SPU. In all three future possible additions, a second pc board must be added making the SPU a 2-wide module.

G. The SPU PROM Simulator

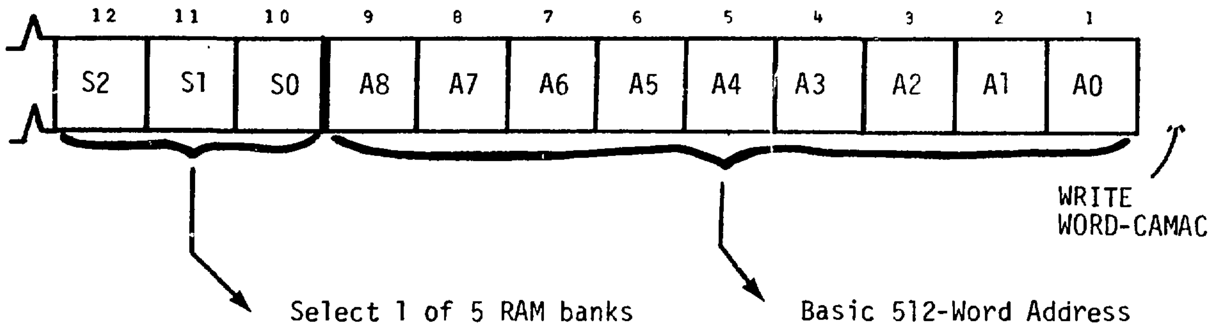
The simulator is a CAMAC module designed to aid in checkout of SPU code. A set of Fairchild 35392 Static RAMs have been arranged in an array of 512 x 40, or five 512 x 8, depending upon the mode of operation. In the former case, the SPU accesses the RAMs over the 20-pin SPU address connector and receives a 40-bit micro-instruction via the 40-pin SPU data connector. In the latter case, a system controller, via the Dataway holding the simulator module, can address and access the RAMs for load-up of SPU code.

The simulator is a 1-wide CAMAC module requiring 1.3 A from the +5 V rail.

CAMAC Command

Subaddress A0

WT1 (F16) Write Memory Address (W12-W1) Register and set CAMAC Access Mode



The select code references the following micro-instruction fields:

$S_2S_1S_0$	100	011	010	001	000
PROM	4	3	2	1	0

WRITE bits W9-W1 then select which word within the selected block will be accessed.

RD1 (F0) Read the contents of the MAR (R12-R1) and clear the CAMAC Access Mode.

Subaddress A1

WT1	Write Memory at MAR, when complete MAR ← MAR+1.
	During BUSY, assert Test Inhibit to SPU.
	W8-W1 are transferred to the selected address at the selected bank.
RD1	Read Memory at MAR, when complete MAR ← MAR+1.
	During BUSY, assert Test Inhibit to SPU.
	R8-R1 contain the contents of the selected address at the selected bank.

The above four CAMAC instructions only will return X to the Crate Controller. The simulator responds to Z·S2 by clearing MAR to zero and clears CAMAC access mode.

Operating Sequence

The user should follow the recommended procedure for loading and testing the simulator memory or some data may be invalid.

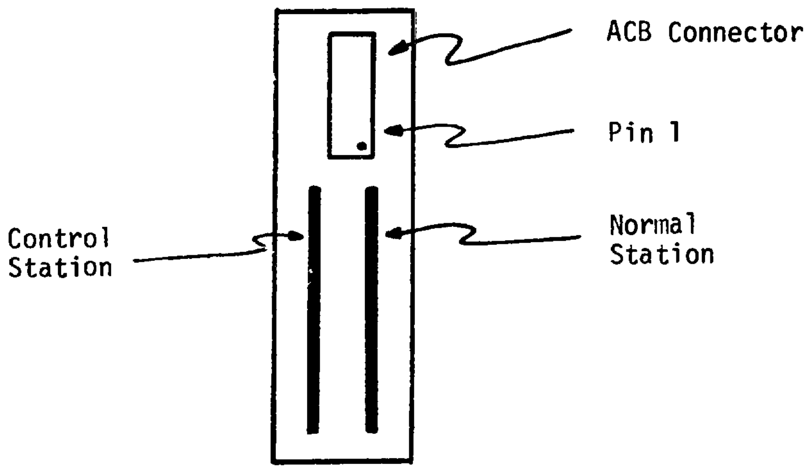
WT1·A0	Set MAR and set CAMAC Access Mode.
WT1·A1	Write data.
.	
.	
.	
.	
WT1·A0	Set MAR to beginning.
.	
.	
.	
RD1·A1	Read and check.
.	
.	
.	
RD1·A0	Read MAR to check and clear CAMAC Access Mode.

The simulator is then ready for access by the SPU. Note that the SPU must run with a 1-MHz crystal in place of the 7-MHz crystal during operation with the PROM simulator. All PROMs must be removed during simulation.

H. Minimum Controller

A 2-wide module has been constructed to take the place of a "normal" crate controller located in crate stations 24 and 25. The minimum controller provides pull-ups for all Dataway and Auxiliary Controller Bus signal lines, and a station number (N) decoder for the Auxiliary Controller Bus (ACB). LAM signals are routed to the ACB connector which is located at the rear of the minimum controller.

The primary use of the minimum controller is in stand-alone operation of a crate with ACC, or expansion of an ACC system to another crate by use of a single control port (see Fig. 3, Section I).



Rear View of Module

III. ACC SOFTWARE

Three questions are usually asked when people consider using the ACC in a system:

1. How do I program the primary processor?
2. How do I transfer code to the ACC?
3. How do I program the SPU?

The first question may be answered immediately in our case--use any PDP-11 computer to create, edit, assemble (or compile), and link code. There are only a few differences in the instruction sets between the various PDP family machines*, and these may be handled by macros or DATA statements.

The second question is somewhat more involved. The code transfer medium can be the following:

1. Down-line load through a branch highway, serial highway (or other compatible interface) through the Dataway Access Port of the ACC. The user must convert his .LDA files to an appropriate data stream to load memory in the ACC.
2. Use a data communications port on the development computer to the terminal port on the LSI-11/2 board. Software must be written on both the development computer and the LSI-11/2 (resides in boot PROM) to handle this transfer technique.
3. The cassette tape unit mentioned in Section I can be loaded with linked code on some other computer system and then read by the LSI-11/2 boot. Software for this unit has not yet been developed, but will be reported on in a later paper dealing with beam line control using the ACC.

The third question is yet more difficult to answer and will be postponed to a separate section in this report (see Section IIIB).

A. The Basic Debug Code

A technique for working with an ACC through a development computer system was used at the Daresbury Laboratory during the engineering of the first ACC. A similar debug technique was put together by J. Potter of the MP-1 staff during the engineering of the LAMPF ACC. The computer uses the RT-11 disk operating system and has a dual floppy disk as a mass storage medium, together with a fully implemented CAMAC crate with parallel branch highway driver. Using the derivative of BASIC for special I/O, CALLBASIC, an ACC Debugger was written (called ACBUG) to allow the ACC user to easily work with the controller and test code during a development session. The load-up and execution of ACBUG is shown in Fig. 5. Note that the position of the Dataway Access Port must be input to the code. Communications with the ACC is via the Branch Driver, CCA-2# or SCC-L2 Crate Controller and the Access Port.

```
RUN BASCAM

BASIC V01B-02
*

USER FNS LOADED

READY

OLD"ACBUG"

READY

RUN

ACBUG     21-SEP-78    BASIC V01B-02

ACC DATAWAY ACCESS PORT SLOT=?8

*SELECT.........RUN        ODT        BOOT       INIT
                RDSTAT     LDMEM      DMPMEM     TSTMEM
*COMMAND........DXRAM      RAMDX

?
```

Fig. 5. Debugger commands.   User Input

The meaning of the ACBUG commands is noted below. A sample session with ACBUG is shown in Fig. 6.

RUN	Enable RUN and issue a RESET to force the LSI-11/2 to 173000_8. This command should be executed when leaving ODT.
ODT	Forces the HALT line to be asserted and transfers control to ODT microcode in the LSI-11/2. If a terminal is connected to the LSI-11/2 port, one can use ODT directly into the 11/2.

#LEEP has ten modified A-1 controllers (CCA-1 1/2) that are compatible with ACCs, in addition to five CCA-2 controllers.

```
 DMPMEM
*DMPMEM...FWA = ?0
          LWA = ?20
*MAR    * MDR **********************************************************
000000  121612  050112  070612  021002  130000  031551  041502  031012
000020  031300
?


 TSTMEM
*TSTMEM...FWA = ?0
          LWA = ?100

*TSTMEM 1 FROM   0  TO  100  COMPLETE*
*TSTMEM 2 FROM   0  TO  100  COMPLETE*
*NO. OF PASSES = 1  TOTAL NO. OF ERRORS = 0

*ENTER 1 FOR CONTINUOUS OR 0 TO EXIT ?0

?


 DXRAM
INPUT FILE NAME ?PTBM
RAM LOAD DONE

?


 RUN

?


 RDSTAT

*MAR = 002402 STATUS = 000000

?


 LDMEM
*LOAD TERMINATES ON D> 177777 OR EX

*ADDRESS = ?0
*MAR = 000000 MDR = 121612  LOAD = ?125252EX
```

Fig. 6. Sample ACBUG session.   User Input

BOOT	Forces a RESET and transfer to 173000_8 for bootstrap purposes.
INIT	Generates a Q-Bus INIT signal to reset all devices connected, with the exception of the 11/2.
RDSTAT	Reads the Status Register in the Access Port and prints the result.
LDMEM	A line-by-line load of memory from a starting address given.
DMPMEM	Reads a given block of memory and prints the result.
TSTMEM	Tests memory over a given address block, zeroes, ones, alternating bit pattern.
DXRAM	Transfer an LDA format file on disk to memory.
RAMDX	Transfer a given address block to disk in LDA format.

B. SPU Microcode

The SPU Control Word, derived from Control Memory (CM) is 40-bits long and provides the user with a detailed level of facility control not available on any single-chip microprocessor or microcomputer. (The LSI-11/2 is microcoded, but the user cannot generally have access to this microcode.) The reader is referred back to Fig. 4, the block diagram of the finite state machine† and Q-Bus interface making up the SPU, for the discussion that follows. In addition, Fig. 7 in this section contains the CM field formats and a mnemonic list of the field definitions useful when coding the machine. Finally, Fig. 8 shows the interim coding sheet and is included at the rear of this section. This rather basic coding form indicates how code is written for the SPU in order that, lacking a cross-microassembler, one can keep track of all the bits!

A future recommendation is to acquire a cross-microassembler* that will operate on a 16-bit minicomputer system. Integral to that, a fuse-link PROM programmer** CAMAC module should be installed with the microcomputer development system such that PROMs can be programmed directly from files generated by the cross-microassembler. Alternatively, these files can be written to the PROM simulator for test and debug prior to programming PROMs.

Testing and initial programming of the SPU was accomplished by a hand programmer and is too prone to errors to be considered the way to proceed in the future (a fresh CM array contains 20K bits waiting to be programmed).

When microcoding the SPU, a mental picture of the entire facility must be kept at hand. The architecture of the machine is both vertical and horizontal, which means that some CM fields are shared by two or more functions, some are directly (unary) assigned, and some fields are coded.

†Using AMD-2910 Sequencer and four AMD-2901A ALUs.
*Available from MICRO-TEC, Sunnyvale, CA., and installed in December 1978.
**Design received from CERN LINAC group in October 1978.

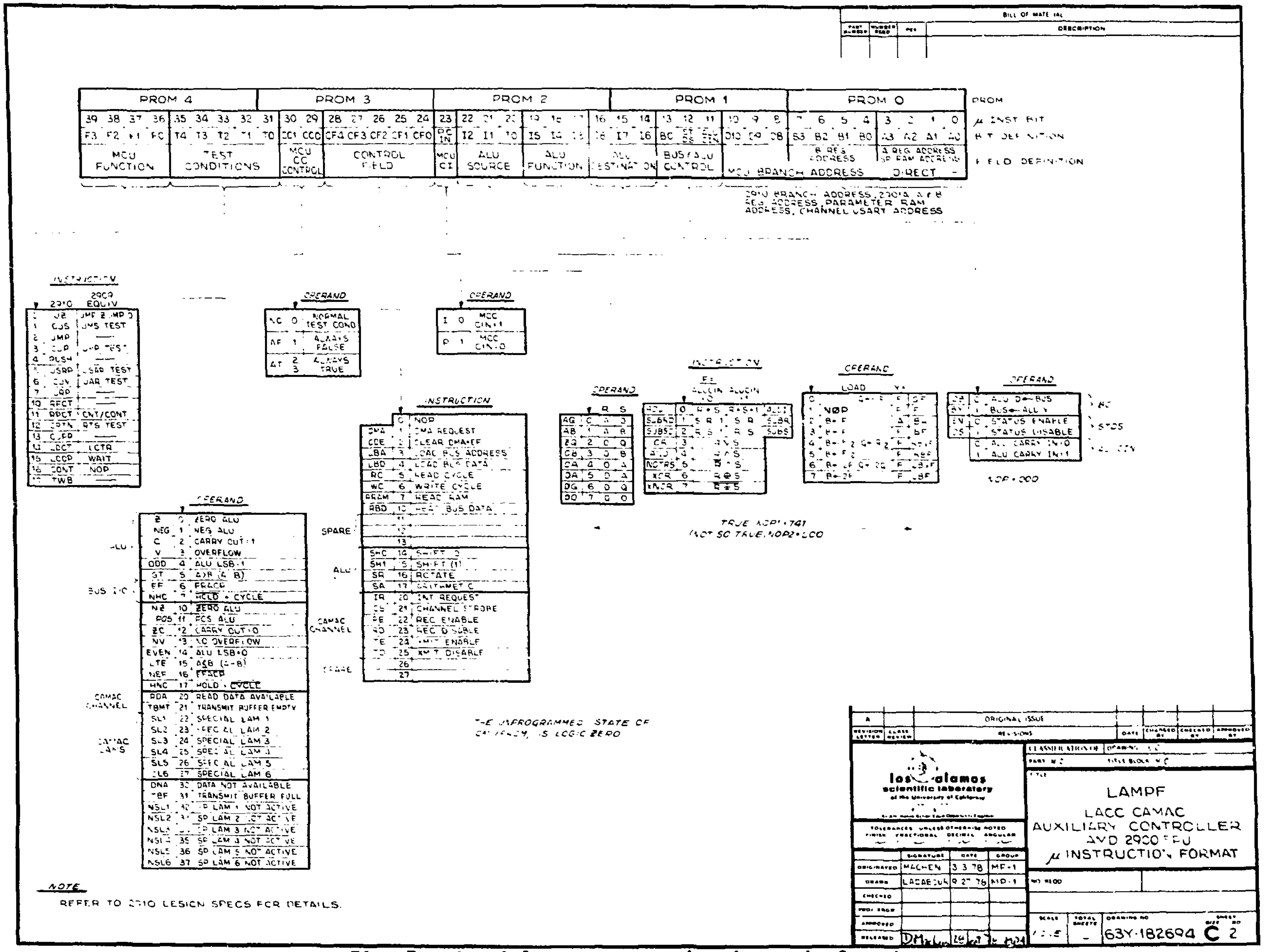


Fig. 7. Special processor unit microcode format.

Coded Fields	Width	Designation
	4-bits	Sequencer Function
	5-bits	Test Condition
	5 bits	Control Field
	9 bits	ALU Source, Function and Destination
Direct Assignments		
	2 bits	Sequencer Condition Control
	1 bit	Sequencer Carry Control
	1 bit	Bus Transfer Direction
	1 bit	ALU Carry Control
	1 bit	Status Latch Enable
	3 bits	D-Input, Sequencer (D10-D8)
Shared Fields		
	4-bits	ALU A Register, Parameter RAM and Sequencer (D3-D0)
	4-bits	ALU B Register and Sequencer (D7-D4)

The sequencer or Micro-Control Unit (MCU) is capable of generating a 12-bit address; we are using only 11 bits in the SPU design. On-board memory accounts for 9 of the 11 possible.

In the discussion to follow, it will be assumed that the reader has a reasonable understanding of the state machine and the bus interface logic of the SPU. The state machine will be considered from the point of view of its microcode, where possible, and not from the point of view of the hardware.

Referring to Fig. 7, a line of microcode might look like:

```
CJP  P  NC  SHO  AB  OR  B   2F  001  B2  A4  ,
```

which would select the ALU registers B = B2 and A = A4, logically OR the contents of B2 and A4, left shift the result with zero fill and store the results in B2. The ALU Y outputs are disabled, the ALU carry-in is set to logic 1, and the status latch is enabled. This instruction will be executed once, and the condition of the ALU output will be tested; execution will proceed at the address in CM formed by the concatenation of (B2,A4) if the ALU output is positive, or continue with the next instruction if the output is otherwise.

The reader must appreciate the somewhat subtle effect of a Pipe-Line Register (PLR) between a CM and the various control word-field destinations. The PLR is clocked on the rising edge of the SPU clock, allowing CM to settle to a new address during the clock period. The MCU is generating this CM address (for the operation to follow the one currently held in the PLR) by special stack operations, selecting other sources or by incrementing the current address (and holding it), or passing the current address to a hold, dependent upon the state of the test condition inputs/MCU carry-in. The source of the next address can be from several different areas (CTR/REG, Direct, and Zero) depending upon the function code, as discussed below.

MCU Function Codes

The instruction field in the high-byte of PROM 4 (F3-F0) is used to instruct the 2910 to select a next address from one of several sources. Generally, a Test Condition modifier will be used with most micro-instructions, and is selected by the next control field (T4-T0) and the condition control bits (CC1,CC0). Either normal condition testing (selected by T4-T0), Always True or Always False, allow complete control of the MCU condition control inputs. Finally, the microprocessor on the MCU can be directly controlled by the carry-in to the microprocessor incrementer (the bit PCIN in PROM 2).

F_3-F_0	MNEMONIC	NAME	REG/ CNTR CON- TENTS	FAIL		PASS		REG/ CNTR
				Y	STACK	Y	STACK	
0	JZ	JUMP ZERO	X	0	CLEAR	0	CLEAR	HOLD
1	CJS	COND JSB PL	X	PC	HOLD	D	PUSH	HOLD
2	JMP	JUMP	X	D	HOLD	D	HOLD	HOLD
3	CJP	COND JUMP PL	X	PC	HOLD	D	HOLD	HOLD
4	PUSH	PUSH/COND LD CNTR	X	PC	PUSH	PC	PUSH	*
5	JSRP	COND JSB R/PL	X	R	PUSH	D	PUSH	HOLD
6	CJV	COND JUMP VECTOR	X	PC	HOLD	D	HOLD	HOLD
7	JRP	COND JUMP R/PL	X	R	HOLD	D	HOLD	HOLD
10	RFCT	REPEAT LOOP, CNTR ≠ 0	≠0	F	HOLD	F	HOLD	DEC
			=0	PC	POP	PC	POP	HOLD
11	RPCT	REPEAT PL, CNTR ≠ 0	≠0	D	HOLD	D	HOLD	DEC
			=0	PC	HOLD	PC	HOLD	HOLD
12	CRTN	COND RTN	X	PC	HOLD	F	POP	HOLD
13	CJPP	COND JUMP PL & POP	X	PC	HOLD	D	POP	HOLD
14	LDCT	LD CNTR & CONTINUE	X	PC	HOLD	PC	HOLD	LOAD
15	LOOP	TEST END LOOP	X	F	HOLD	PC	POP	HOLD
16	CONT	CONTINUE	X	PC	HOLD	PC	HOLD	HOLD
17	TWB	THREE-WAY BRANCH	≠0	F	HOLD	PC	POP	DEC
			=0	D	POP	PC	POP	HOLD

*If condition fails, then hold register. If condition passes, then load register with (D,B,A).

The table (reproduced and modified from that available in the 2910 literature) details what control each MCU function has over the on-chip facilities. A brief description of each function code follows:

Command 0 - JZ - Jump Zero is used to jump to the beginning of CM. When used at the end of a program (or when a reset is applied), the command unconditionally sets the next address to be zero. To remain at location zero while testing a selected condition, PCIN = 1.

Command 1 - CJS - Conditional Jump to Subroutine via the address in (D,B,A). If the condition is true, the next address in sequence is pushed onto the stack and the program jumps to the address specified in (D,B,A). For a false condition, the next address in sequence is selected.

Command 2 - JMP - Unconditional jump to address in (D,B,A).

Command 3 - CJP - Conditional instruction which, if true, selects (D,B,A) as the next address. Very similar to Command 1 except that the next address in sequence is not pushed onto the stack.

Command 4 - PUSH - Conditional instruction which pushes the next address in sequence onto the stack and if the condition is true, the counter is loaded from the PLR branch address field. If false, the counter is not loaded.

Command 5 - JSRP - Conditional Jump to Subroutine via PLR address or register address. A push onto the stack is always performed, and if the condition is true, the register contents will be used as the next address. If false, the PLR branch address field will be used as the next address. The register must have been loaded with the correct value for the subroutine address by the instruction prior to the Conditional Jump to Subroutine.

Command 6 - CJV - Conditional Jump Vector instruction is used to load the SPU with the address from the CSR when the External Function bit (bit 15) is also set to logic 1. This instruction is placed at address 0, and until an External Function is received, the condition will remain false. If, at the same time, PCIN = 1, the address will not be incremented and the program will hang at address 0. When an External Function occurs, the condition will become true and the program will go to the address supplied by the CSR address field. (See Section IIF on SPU hardware design and operation.)

Command 7 - JRP - Conditional Jump instruction via the contents of the Register Counter, or the PLR branch address field. This is similar to Command 5 except that no push onto the stack occurs.

Command 10 - RFCT - Repeat Loop, Counter ≠ Zero instruction assumes that a count has been loaded into the Register/Counter. If the count is not zero, the count is decremented by 1 and the address of the next micro-instructrion taken from the stack. When the count is zero, control passes to the next sequential micro-instruction and the stack is pop'ed.

Command 11 - RPCT - Repeat Pipeline Register, Counter ≠ Zero instruction is similar to Command 8 except that the branch address in the event of the count ≠ 0 comes from the pipeline register rather than the stack. Also, a pop of the stack is not performed.

Command 12 - CRTN - Conditional Return from Subroutine instruction will return to the main program by completing a pop on the stack if the condition is true. If false, the next micro-instruction in sequence in the subroutine will be performed.

Command 13 - CJPP - Conditional Jump Pipeline Register and Pop stack instruction. If the condition is false, the program will sequence onto the next micro-instruction. If true, the program will jump to the address supplied by the PLR branch address field, and the stack will be pop'ed.

Command 14 - LDCT - Load Counter and Continue instruction allows the counter register to be loaded with an 11-bit number.

Command 15 - LOOP - Conditional Test End of Loop instruction causes the next instruction in sequence to be executed if the condition is true. If the condition is false, the microprogram will loop back via the address held on the top of the stack file. This instruction would normally be placed at the end of a loop.

Command 16 - CONT - Continue instruction which causes the microprogram to increment to the next micro-instruction in sequence. A NOP.

Command 17 - TWB - Three Way Branch instruction which assumes that the register/counter has been preloaded and a branch address has been pushed into the stack. The instruction decrements the count and branches to the address given by the stack as long as the counter is greater than zero. When the counter = 0, the next address is selected from the PLR branch address field. However, if at any time the condition code becomes true, the microprogram counter, (i.e., pc), will be the new address selected. The stack is automatically pop'ed whenever the count reaches zero or the conditional test becomes true.

Test Conditions

The selectable Test Conditions (TC) relate to fixed terms (true and false for each), ALU generated terms, or bus I/O generated terms. TC(20), TC(21), and TC(30),TC(31) are reserved for future possible use with a communications channel controller. TC(0)-TC(5) and TC(10)-TC(15) are ALU terms and self explanatory, while TC(6),TC(7) and TC(16),TC(17) require more detail. The latter conditions refer to the External Function control bit in the CSR and the DMA hold/cycle-in-progress signals respectively, from the bus I/O. By using these TCs, all possible states of the bus, of concern to the SPU, may be tested.

TC(23)-TC(27) and TC(33)-TC(37) allow the state of the six Special LAMs from the ACC Control Port to be tested. Finally, the STDS control bit (in PROM 1) controls the enable/disable state of the TC staticizer. TCs are latched on the rising edge of the SPU clock (the beginning of a cycle); the TCs may be held from the last cycle by letting STDS = 1 in the current and subsequent cycles.

Condition Control

The user is able to determine the effect of the test conditions on the MCU through bits 30 and 29 of the micro-instruction; CC1 and CC0 can take on values that define either "Normal Conditions" - NC (0), "Always True" - AT (1), or "Always False" - AF (2 or 3). This field is useful when used with the conditional MCU instructions, as shown in the MCU table of instructions.

Control Field

The Control Field provides the SPU with control over all bus I/O options, shift multiplexer function, and future options.

The shift function controls (CF(14)-CF(17)) have been repeated every four decoded commands, but are unique for commands 14-17. It is possible then to execute a shift function while performing some other control.

	CF(i)	Function
NOP	CF(0)	- No control action; ALU shift logic set for shift logical with zero fill.
DMA	CF(1)	- Request DMA hold on ACC bus (tested with TC(7), NHC or TC(17) HNC). Shift logic: Logical, 1 fill.

	CF(i)	Function
CDE	CF(2)	- Clear DMA hold and External Function bit in CSR. Shift logic: Rotate.
LBA	CF(3)	- Load the Bus Address Register from ALU outputs; Shift logic: Arithmetic.
LBD	CF(4)	- Load the Bus Data Register from ALU outputs; Shift logic: Logical, 0 fill.
RC	CF(5)	- Request Read cycle on ACC bus; use contents of BAR as address, data to BDR. Shift logic: Logical, 1 fill.
WC	CF(6)	- Request Write cycle on ACC bus; use contents of BAR as address, data from BDR. Shift logic: Rotate.
RRAM	CF(7)	- Read contents of SPU parameter memory, at address in A-Field, into ALU direct inputs (BC = 0). Shift logic: Arithmetic.
RBD	CF(10)	- Read contents of BDR into ALU direct inputs (BC = 0). Shift logic: Logical, 0 fill.
-	CF(11)	- Spare; may be used for multiplier option.
-	CF(12)	- Spare; may be used for multiplier option.
-	CF(13)	- Spare; may be used for multiplier option.
SHO	CF(14)	- Logical shift ALU, zero fill.
SH1	CF(15)	- Logical shift ALU, 1 fill.
SR	CF(16)	- Rotate ALU.
SA	CF(17)	- Arithmetic shift ALU.

CF(20) - CF(25) are reserved for the future communication channel controller.

CF(26) and CF(27) are spare and may be used with the multiplier option.

ALU Instruction Field

(I_2, I_1, I_0)	ALU source operand. A, B, D, Q, and zero refer to selected registers within the ALU. A and B are selected by the A-field and B-field of the Control Word, and refer to a multi-port 16 x 16 RAM in the ALU. D is the ALU Direct inputs, while Q is an additional working register within the ALU. Zero is truly a 16-bit word containing zeros.
(I_5, I_4, I_3)	ALU operation code (see Fig. 7 for list of possible arithmetic and logical operations).
(I_8, I_7, I_6)	ALU Destination operand and output control. Register modification, storage, and shift-logic direction is controlled by this field.

Other Micro-instruction Fields

BC	Control for SPU bus-to-ALU Output or D-Input. 0 = ALUD ← BUS 1 = BUS ← ALUY
ALUCIN	ALU carry control for arithmetic operations. 0 = Carry in = 0 1 = Carry in = 1
STDS	Test condition latch disable. This control allows one to hold the status of some past operation that generated a set of test condition terms. DS = 1 disables the latch.

The use of the D, A, and B- Fields have been explained in the contest of specific micro-instructions listed above.

PCIN	Controls the state of the MCU carry-in. I: PCIN = 0 = increment address P: PCIN = 1 = pass address

Some Notes
Location 0 of Control Memory must have the following instruction:

```
CJV  EF  NC  NOP  P  NOP  000  --
```

while a RESET or end-of-code instruction is:

```
JZ  NOP  NC  CDE  P  NOP  000  --  .
```

The ALU and (D,A,B) fields could contain other instruction, as needed. Note the use of an MCU Address Pass Command in both of the above instructions. Failure to follow this convention will cause address 1 to be executed in error.

PAGE ____

	LACC AM2900 SPU INSTRUCTION												4	3	2	1	DIRECT BRANCH 0
			TC		MCU	ALU							MCU FUNCTION | TEST	Cond Cont | CONTROL	MCU | SRC | FNC	DST | | AU | MCU	B-REG | A-REG SP-RAM
ADDRESS	MCU FNC	TEST	MUX	CONTROL	C_i	SRC	FNC	DST	BC	ST	C_i	D-B-A	F_3 F_2 F_1 F_0 T_4 T_3 T_2 T_1	T_0 CC_1 CC_0 CF_4 CF_3 CF_2 CF_1 CF_0	C_i I_2 I_1 I_0 I_5 I_4 I_3 I_8	I_7 I_6 BC ST C_i D_{10} D_9 D_8	B_3 B_2 B_1 B_0 A_3 A_2 A_1 A_0

Fig. 8. Interim coding sheet.

APPENDIX

The following two technical notes were generated at the time the beginning ideas of The Auxiliary Controller were being formulated. I feel it is useful to include them in this report, as they bear directly on the LAMPF ACC. Only the communications controller has not been cast in hardware, although a LSI circuit* has been found that will do the job. The concept of multiple high-speed firmware subroutine facilities has been established in both bench-mark and field testing and seems to be generating a great deal of interest in the experimental physics area at LAMPF.

*COM5025 from SMC Microsystems Corporation, 35 Marcus Blvd., Hauppauge, NY 11787; $24 from Century Electronics.

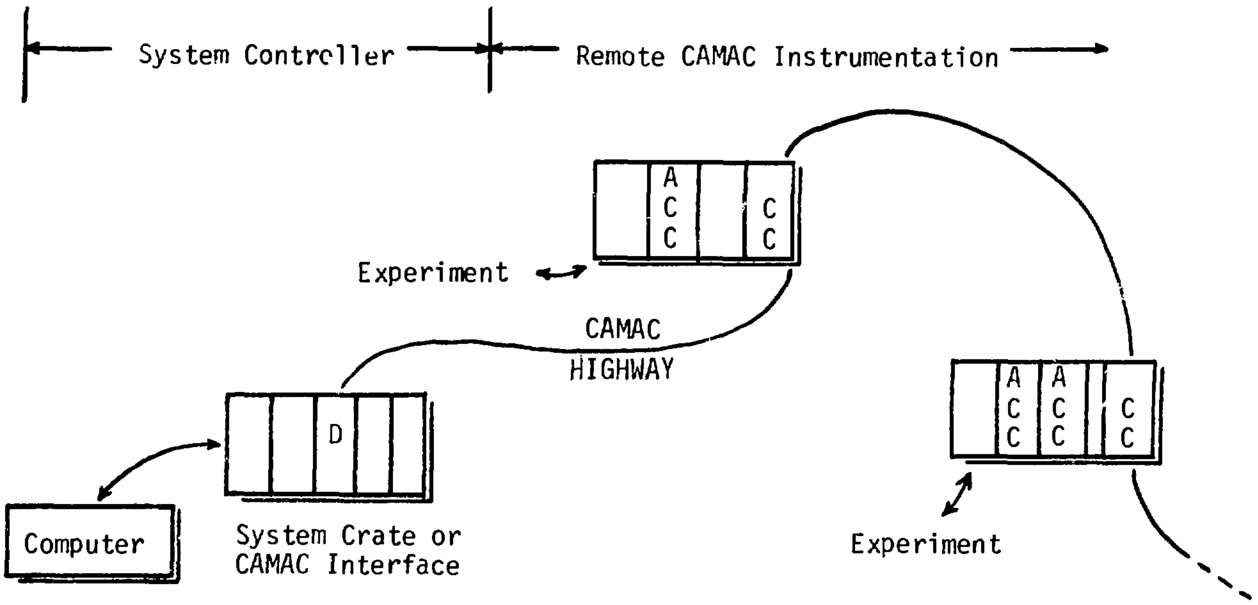
CAMAC AUXILIARY CONTROLLER CONCEPTS[†]

Recent discussions, prompted by the probable need for fast in-crate ADC or TDC processing at Daresbury Laboratory, have generated some interesting ideas in the area of Auxiliary Crate Controllers (ACC). This note begins an effort to tie these ideas together, with the desired outcome being a workable design approach towards a powerful in-crate processing system.

Primary Design Aim

Distribute processing tasks remotely so that:

1. System computer overhead is reduced, and
2. The data rate capabilities of the CAMAC dataway can be realized, if needed.



The CAMAC dataway can be effectively shared* by the CC (conventional Crate Controller) and one or more ACCs so that processing tasks can be placed within the crate. This arrangement is a logical extension of data-link communications to satellite mini computer-driven CAMAC crates, but takes advantage of the powerful microprocessor technology available to us currently.

What, then, does the ACC look like and how can it be arranged? A further question is, how can it best communicate with the system computer--other than via the CAMAC Highway (parallel or serial)?

ACC

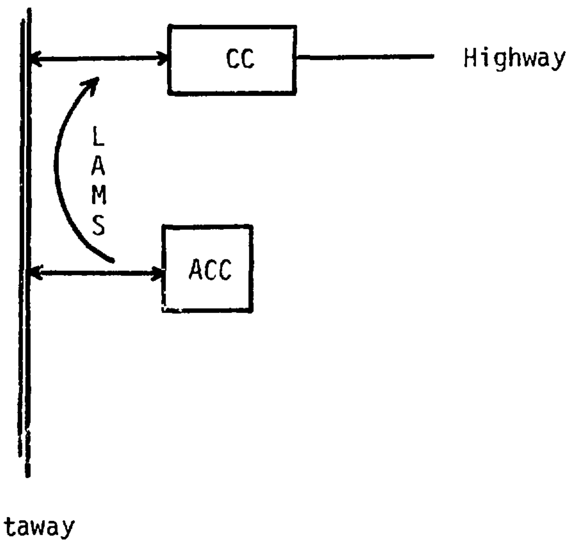
- One can be safe in saying that the ACC must be able to cope with normal transfers and Request/Grant or ACL priority arbitration for dataway control.
- Further, the ACC should be able to execute a fast CAMAC cycle ($\sim$200 ns), and respond, when addressed, to some specified CC-generated CAMAC functions.

†Daresbury Laboratory Note CSE 3-76, D. R. Machen.
*DOE/EV-0007 "Multiple Controllers in a CAMAC Crate.

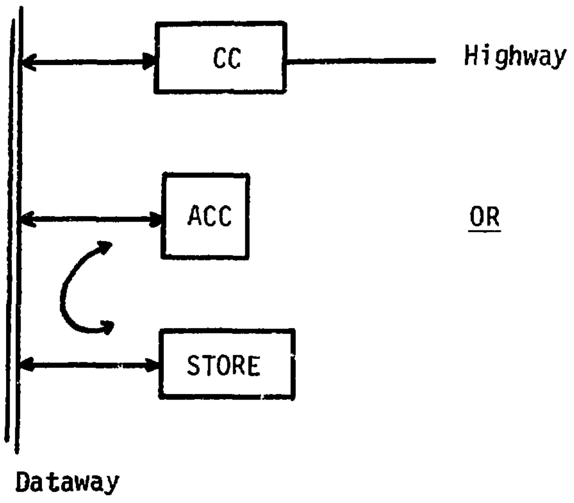
- The ACC needs to have both RAM and PROM store available, but not necessarily on the same physical board.
- The ACC can be implemented with either a single-chip microprocessor (slower) or a bit-sliced microprogramming facility (faster).

Communications with System Computer

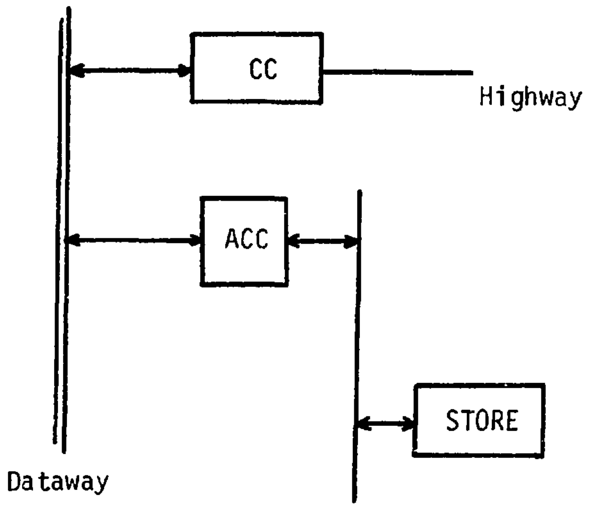
- An ACC can utilize conventional LAM signaling to indicate some need to communicate with the System Computer:



- However, the ACC is processor-based and, therefore, will have associated with it an internal bus. Communication between the ACC and its internal elements (store, for example) can either pass along this bus or (through other dataway time-sharing techniques) along the dataway.

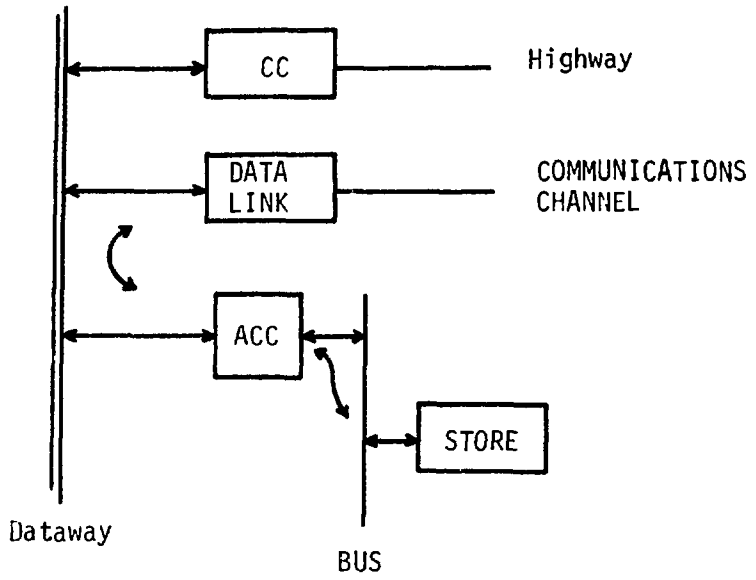


OR

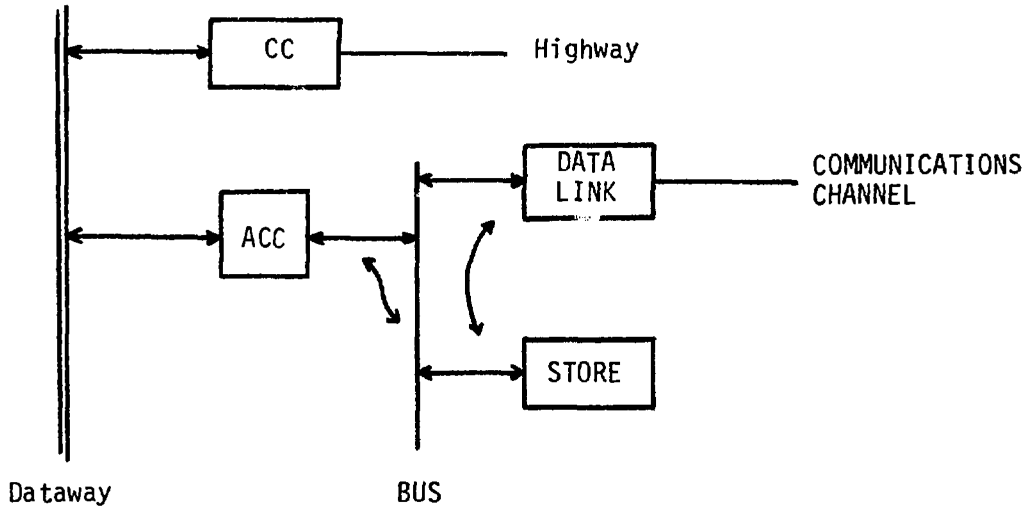


A logical extension of these ideas involves alternative communication paths to the system computer.

- As the ACC is able to exercise control over the dataway, one can install CAMAC data link (Daresbury or LAMPF-style data link) modules in the crate and operate the distributed system much like a satellite mini configuration.

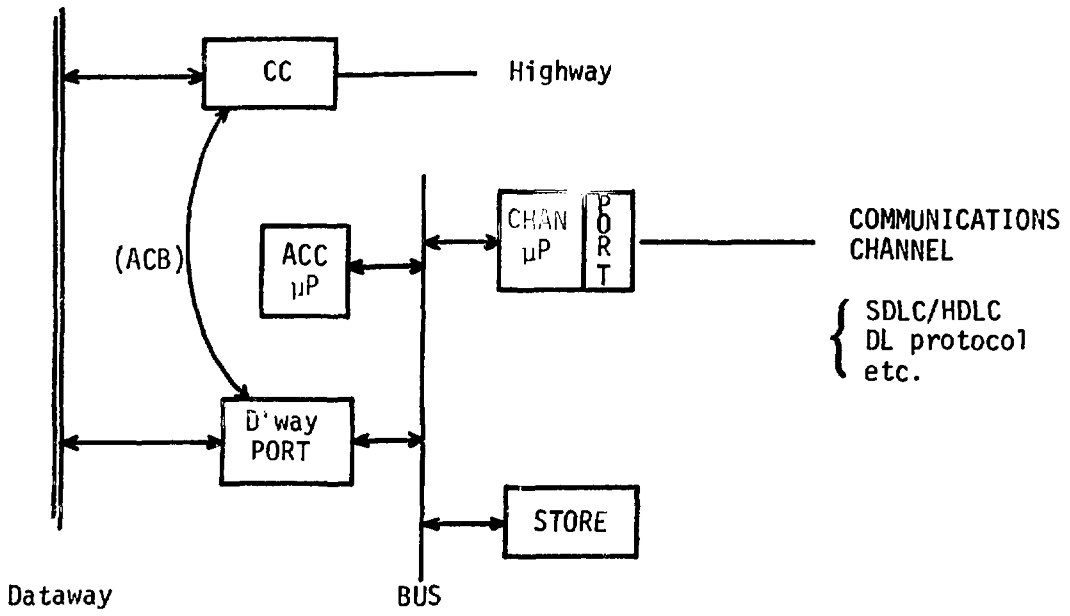


Another approach to the communication channel link exploits the processor (ACC) internal bus. The link obtains block data directly from the ACC store.



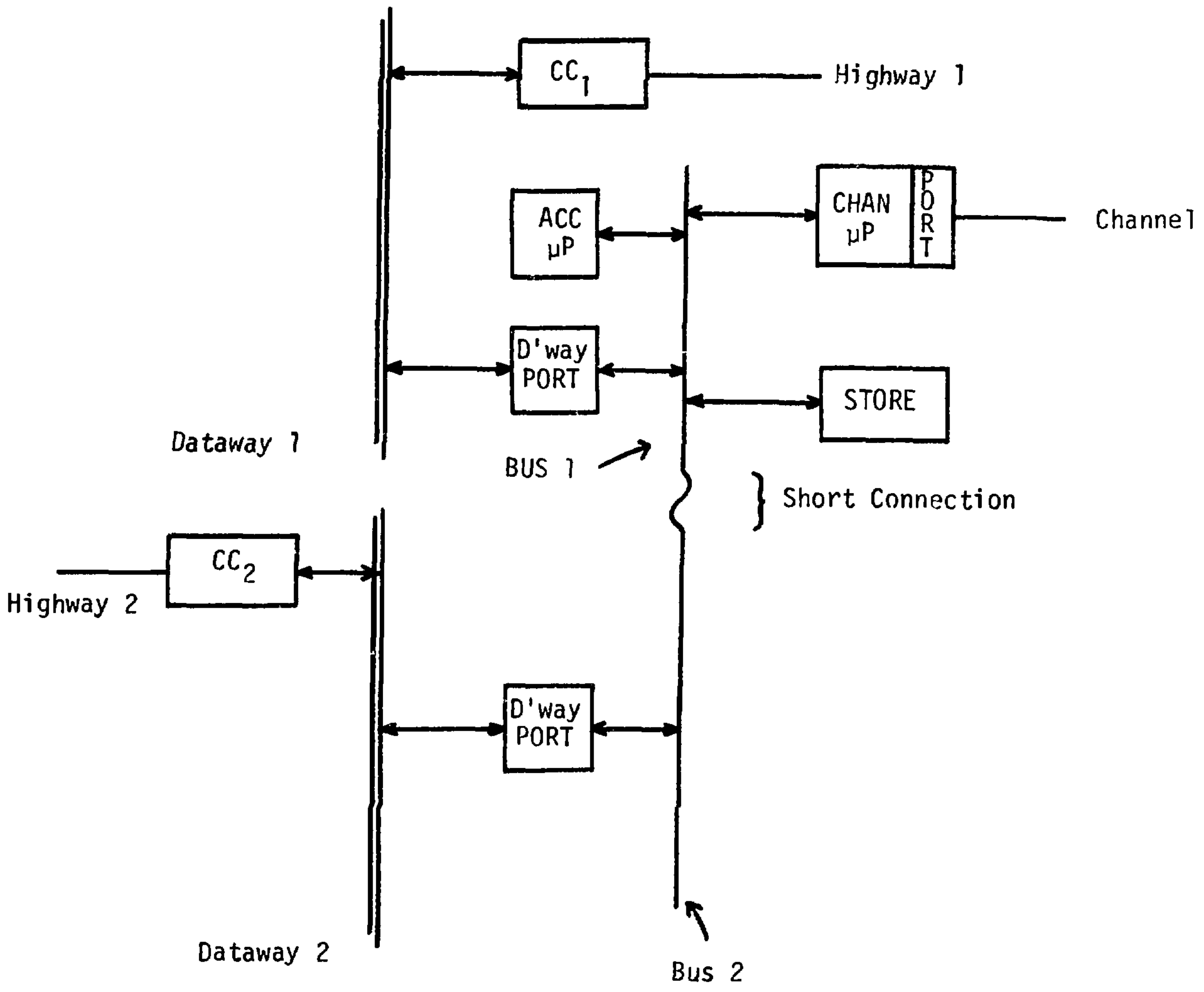
- In the former method, the ACC must not only perform the in-crate processing task but must also provide information and protocol control to the data link module. The latter method eliminates one redundant path and allows ACC and data link to function in parallel (if the data link were intelligent enough to DMA information from/to the store, and control itself in addition).

- The data link can, if microprogrammed and given the appropriate port connection, generate serial highway protocol, SDLC/HDLC protocol, Daresbury Laboratory protocol, or any other communication channel protocol. Such a link implies a built-in processor similar to the ACC.



What About Crate-Crate Auxiliary Control?

One ACC in a crate cluster could have access to modules in two or more crates through the ACC internal bus.



System considerations and microprocessor speed requirements could well limit the usefulness of this scheme. A complete ACC system in each crate may be desirable in any case.

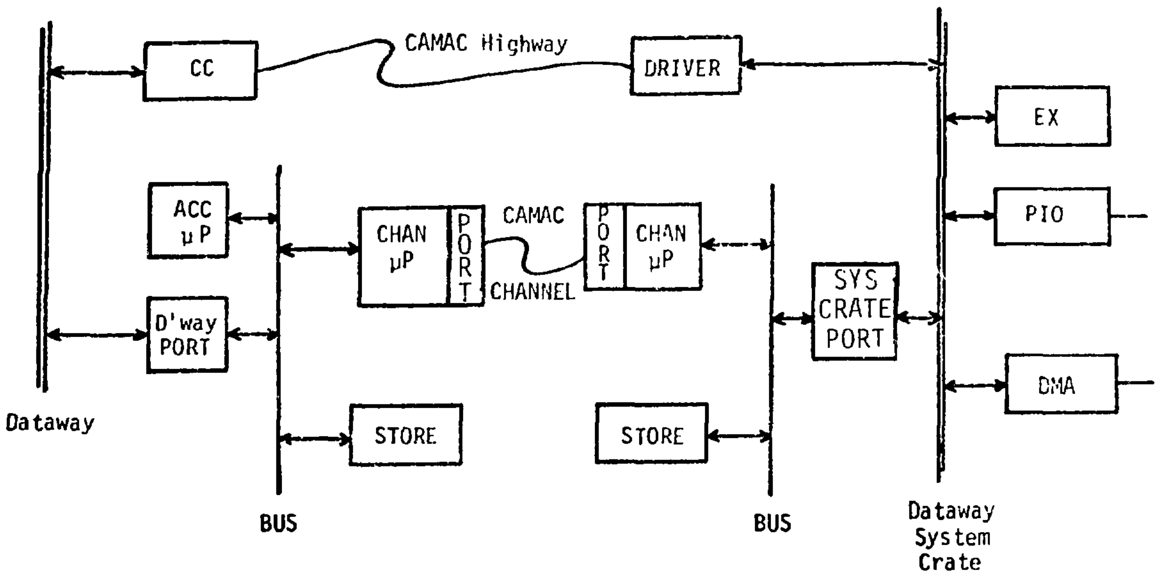
Other Uses of the ACC

There is no fundamental reason why the ACC configuration presented in this note cannot be considered a bilateral one.

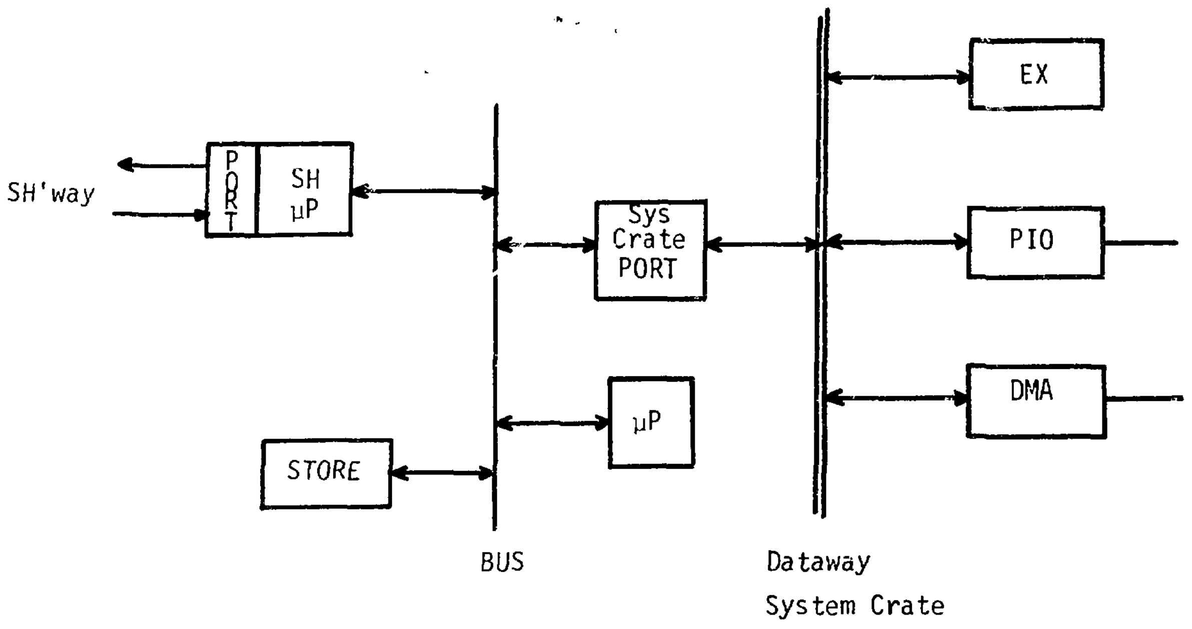
- Assume we have a system crate driving a CAMAC Highway in addition to providing both programmed and DMA interface to the system computer. Further assume that the crates connected to the Highway contain ACCs as described in this note.

The CAMAC "channel" from each ACC is capable of sending along block-oriented data in contrast to the Highway, which is "single-action" oriented. How, then, do we convey this information to the system computer with the least overhead?

Comm. Channel	Receiving Device
SH	Special SD with block read/write protocol
Other DL	An ACC configuration, interfaced to system crate
HDLC/SDLC	HDLC/SDLC channel port (see note 4-76)

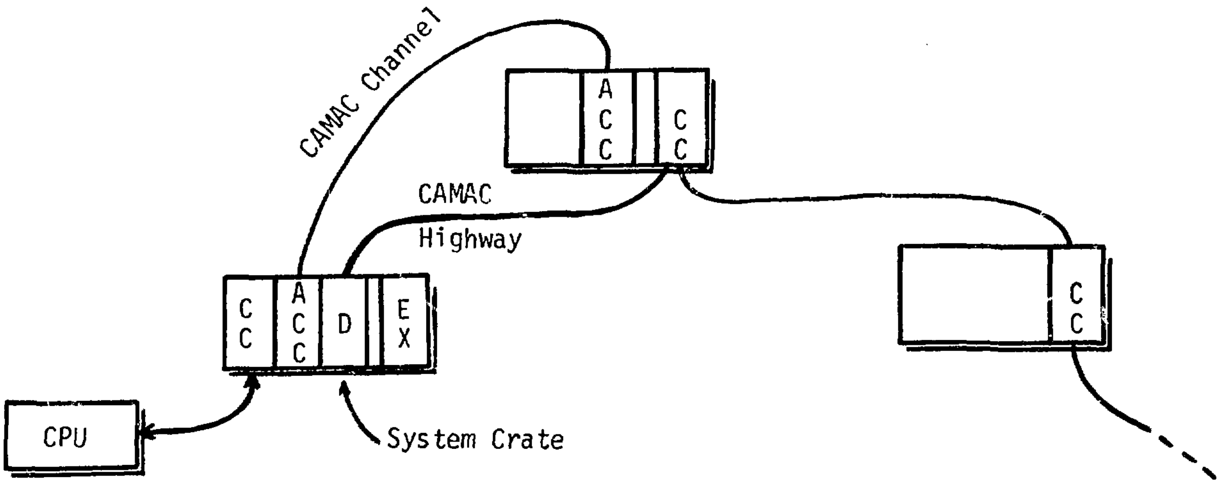


Further, the ACC with system crate interface could be used as a Serial Driver, provided the maximum performance of the Serial Highway was not desired.



The use of the ACC in a system crate implies direct transfer of information between the ACC and the system computer's memory, under control of the DMA controller.

It is conceivable that the DMI* controller could be replaced or significantly modified by such an ACC-oriented scheme!

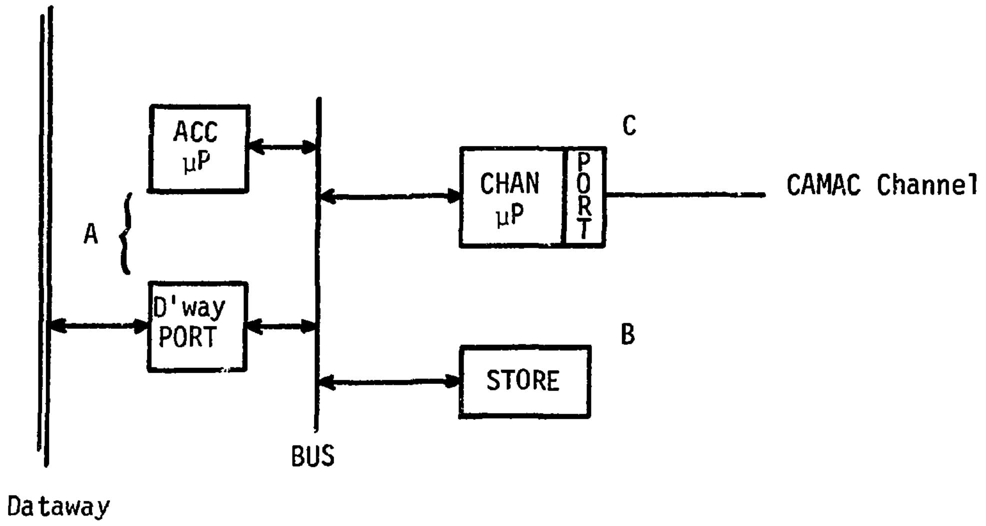


CAMAC channel information can be both data or macro instructions to ACC main processor imbedded within a message structure protocol (SH, SDLC/HDLC, DLDL, etc.).

*DMI - Direct Memory Increment.

CONVENTIONAL CAMAC SYSTEMS AND THE ACC†

Given the ACC components shown below as a minimal set of components to accomplish the auxiliary controller function, we must ask how the device fits into existing CAMAC structures and what it could do that is new.



The ACC processor/Dataway port (A), store (B), and channel processor/port (C) seem to be the lease one should assemble to create an ACC.

Perhaps the major point in favor of such an arrangement in a CAMAC crate with a conventional crate controller (CC) is the mode of control that would be possible.

Conventional CAMAC systems and CCs (see Fig. 1) communicate between CC and computer in a single-action mode; each CAMAC function to be executed must be passed to the CC one at a time along the I/O bus or highway (DMA controllers do not, but they are rather special-purpose).

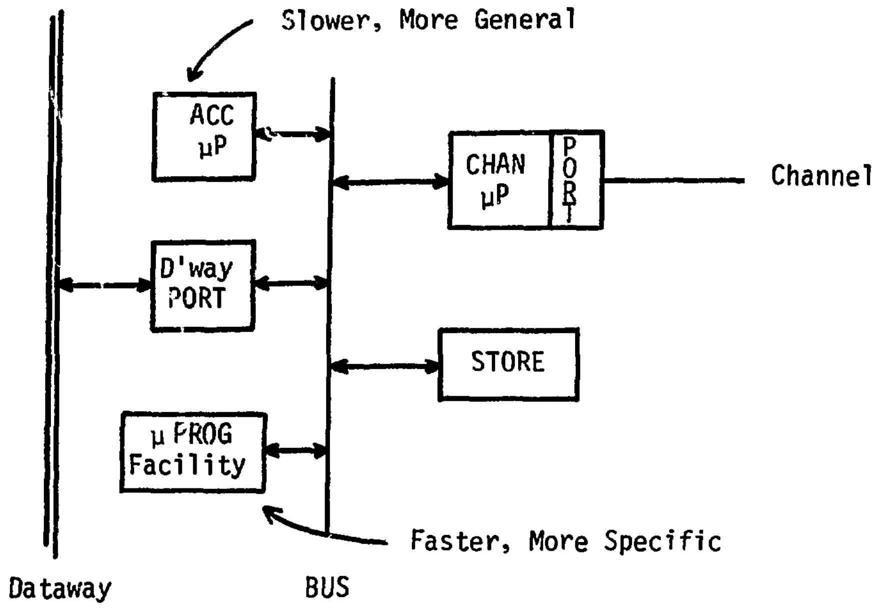
The ACC, with channel port, allows a CAMAC channel to communicate a message block between the crate and the central computer. The message blocks may contain data to or from the computer, or it may contain a sequence of CAMAC instructions, or it may contain instructions to carry out some preprogrammed task. The data link scheme used at DL and LAMPF provides the same, except that one redundant path would be eliminated with ACC. The ACC, with or without the channel processor port, can serve to distribute tasks away from a central computer (much the same way as satellite minis do now). The difference is physical size and capability in an ACC without the channel port.

Capability

A 16-bit word length ACC processor seems reasonable. Further, it would be an added feature to have available special CAMAC-oriented functions in microcode. One should not, however, try to build a computer or duplicate the general purpose computer instructions in a microcoded processor.* One proposal would be to provide a micronova TI 9900, or Ferranti F100L as the general purpose

†Daresbury Laboratory Note CSE 4-76, D. R. Machen.
*Microcoded processor or microprogramming facility, constructed with bipolar, bit-sliced chips.

"provider" of instructions, and a companion facility, on the internal bus, that was capable of executing special and/or high-speed functions.



The microprogramming facility can be plugged in, or not, depending upon the task to be performed.

Figures A1 to A4 take each common CAMAC configuration with an ACC in an attempt to sort out the question of what does it buy us.

Addition of an ACC to (1) or (2) in Fig.A1 does not seem to gain one much. Other configurations, however, have more appeal. Systems shown in Figs. A2 to A4 could provide a new degree of freedom to CAMAC instrumentation.

Figures A5 and A6 indicate the possible organization of the ACC and system configuration, respectively. One must ask, what the messages to ACCs will contain, assuming that they are imbedded within a SDLC/HDLC message protocol? Shifting our thinking from Branch Highway and Serial Highway multicrate configurations to CAMAC channel configurations, may well allow CAMAC to be associated with some form of industry standard I/O bus, thus reducing the problems involved with computer-dependent drivers that exist today.

As I mentioned earlier in this note, the big advantage in moving to intelligence in the channel port and CAMAC controllers is the possibility it gives us to eliminate single-action commands and incorporate multiple-action commands. This is evident when we answer the question concerning the content of a CAMAC channel message.

1. Machine code for ACC (down-line loading ACC)
2. Single-action NAF
3. String of NAFs and data
4. String of data at single NAF
5. Higher level source to ACC; resident interpreter/compiler
6. Task execution
7. String of tasks

On reflection, it is evident that No. 2, only, is common with present highway configurations. Further, by distributing the intelligence throughout a system we have moved the computer-to-dataway mapping away from the driver and into the crate, allowing a more sophisticated set of message possibilities along a more standard communications channel.

Multi-crate Connections

Multi-crate connections may be accomplished either through internal ACC bus extensions within crate clusters or through store and forward message handling at each ACC channel port. The latter has not been investigated in depth at this point, but seems to be one solution, based upon computer network practices. The lack of implicit intelligence in SCC led us to a unidirectional loop for the Serial Highway. With a message-oriented scheme, we should be able to do much better by using techniques developed for data communications networks.

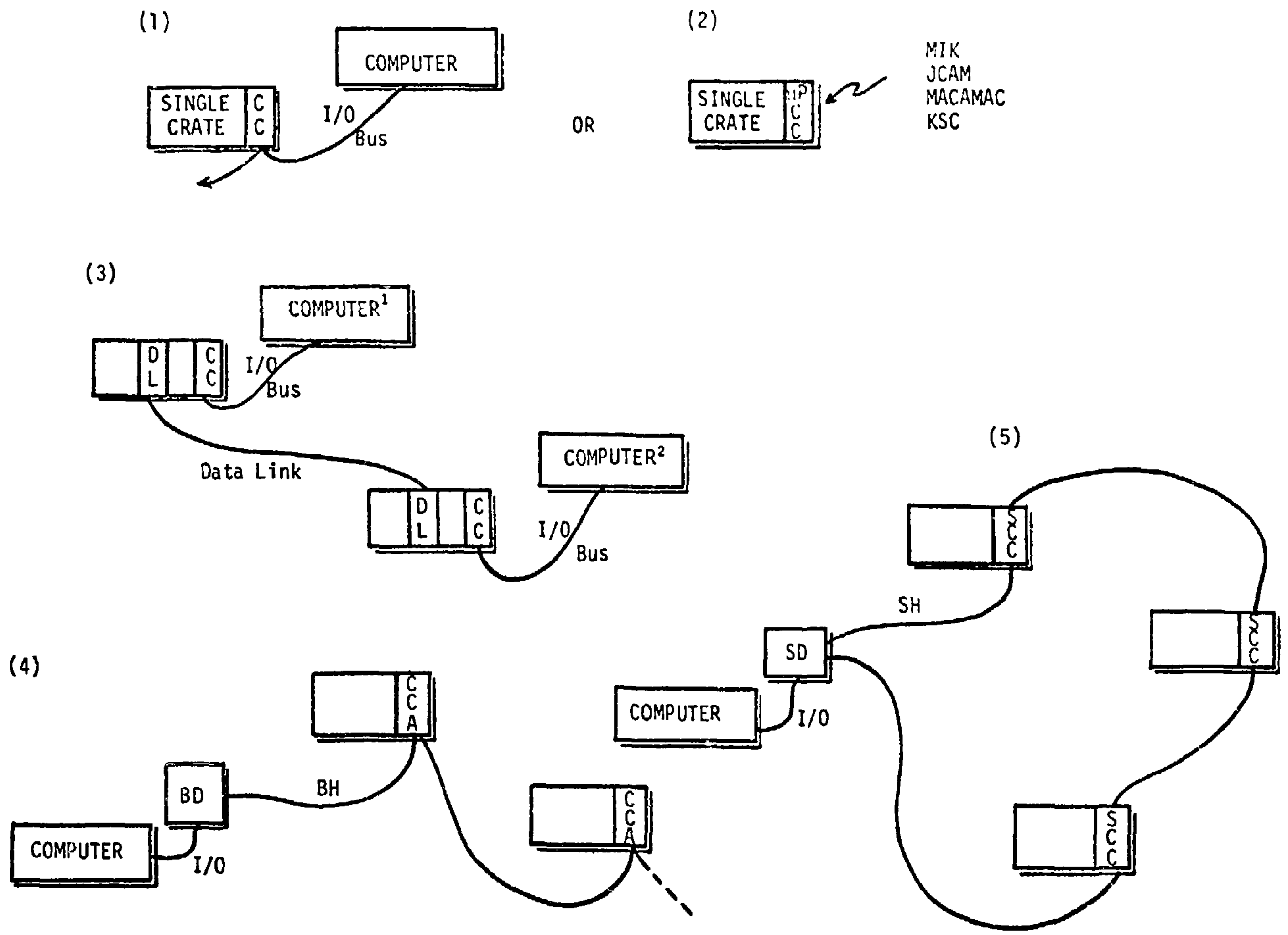


Fig. A-1. Typical CAMAC configurations.

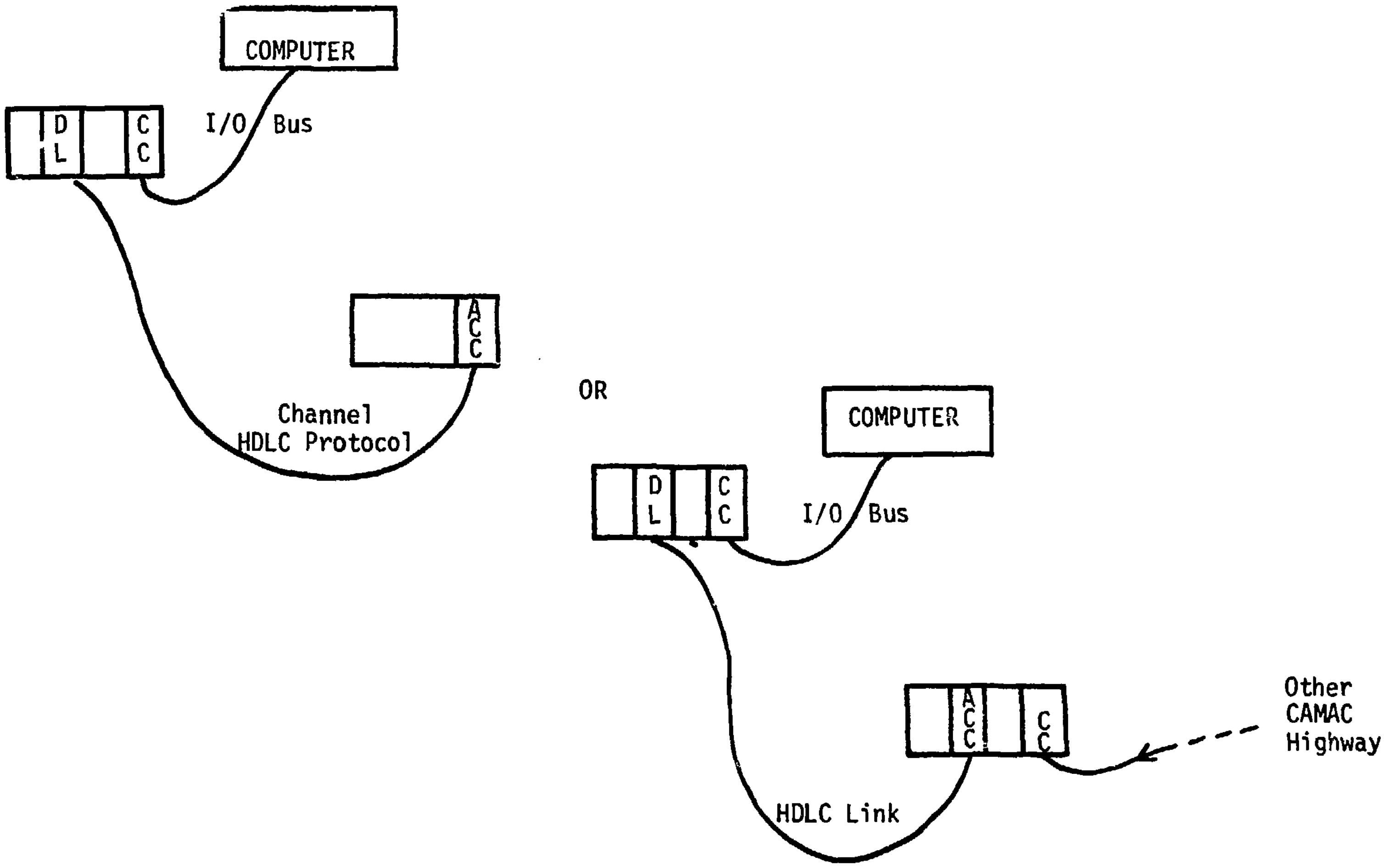


Fig. A-2. Data link coupled systems.

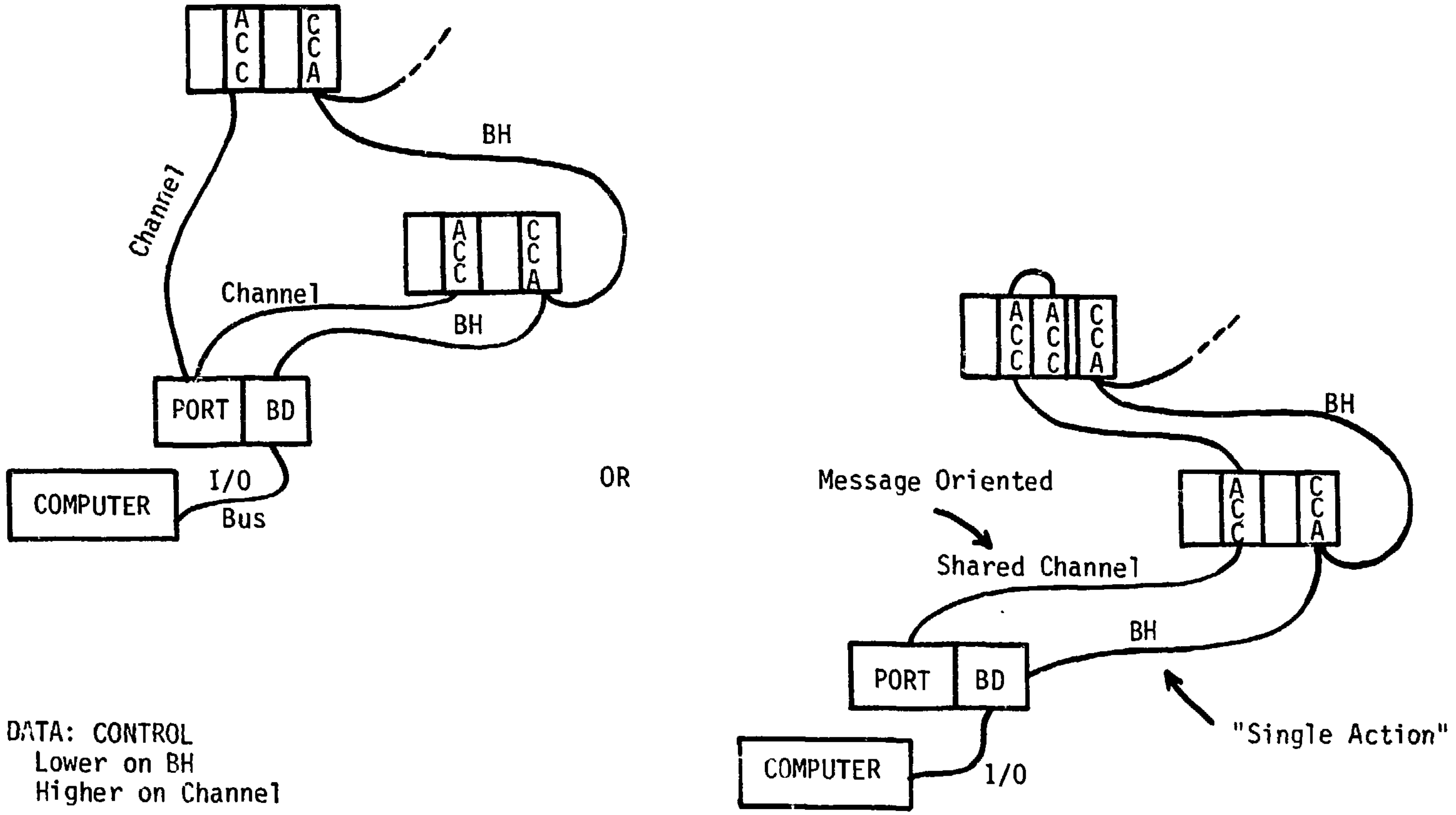


Fig. A-3. Branch highway systems.

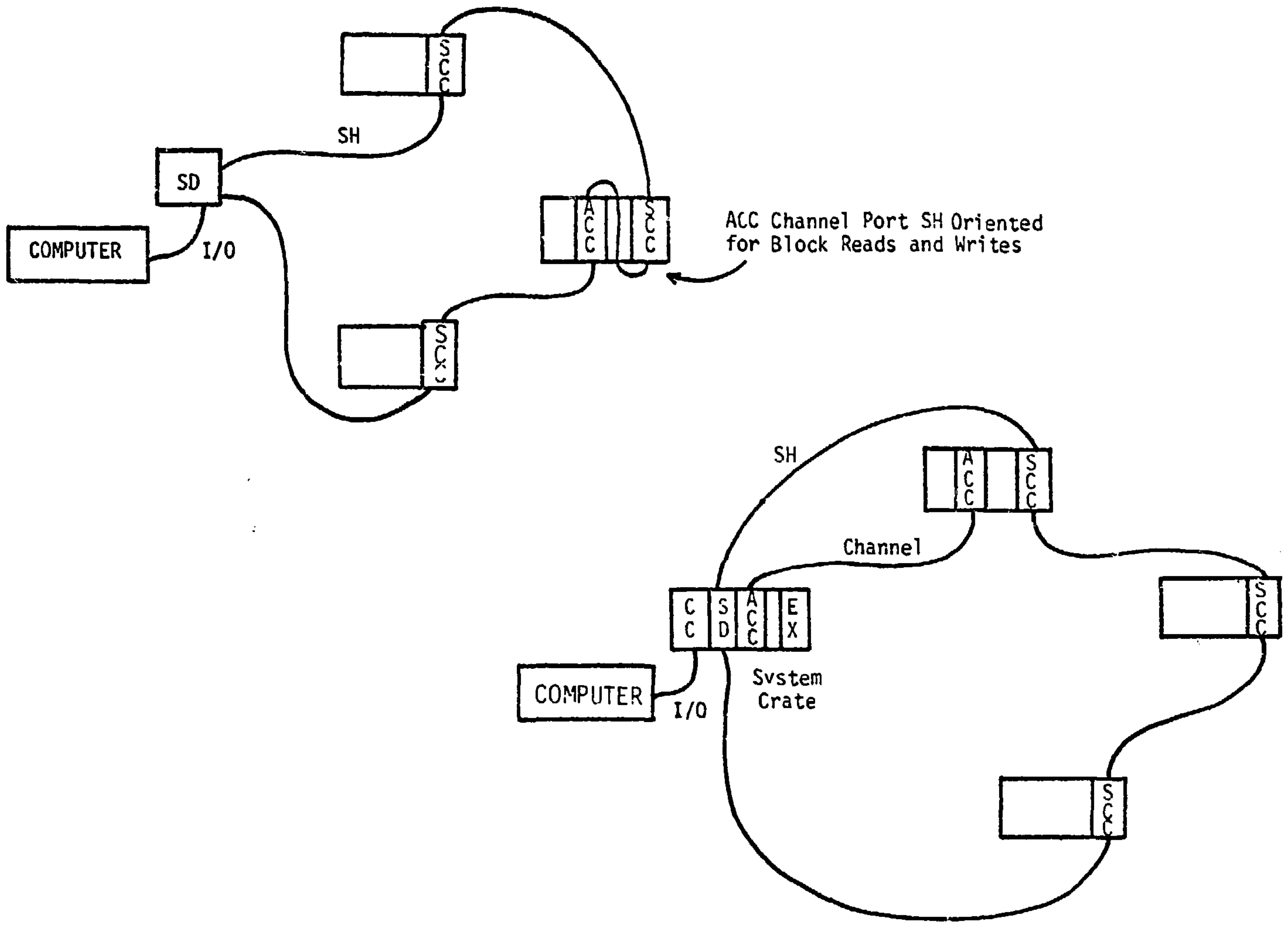


Fig. A-4. Serial highway systems.

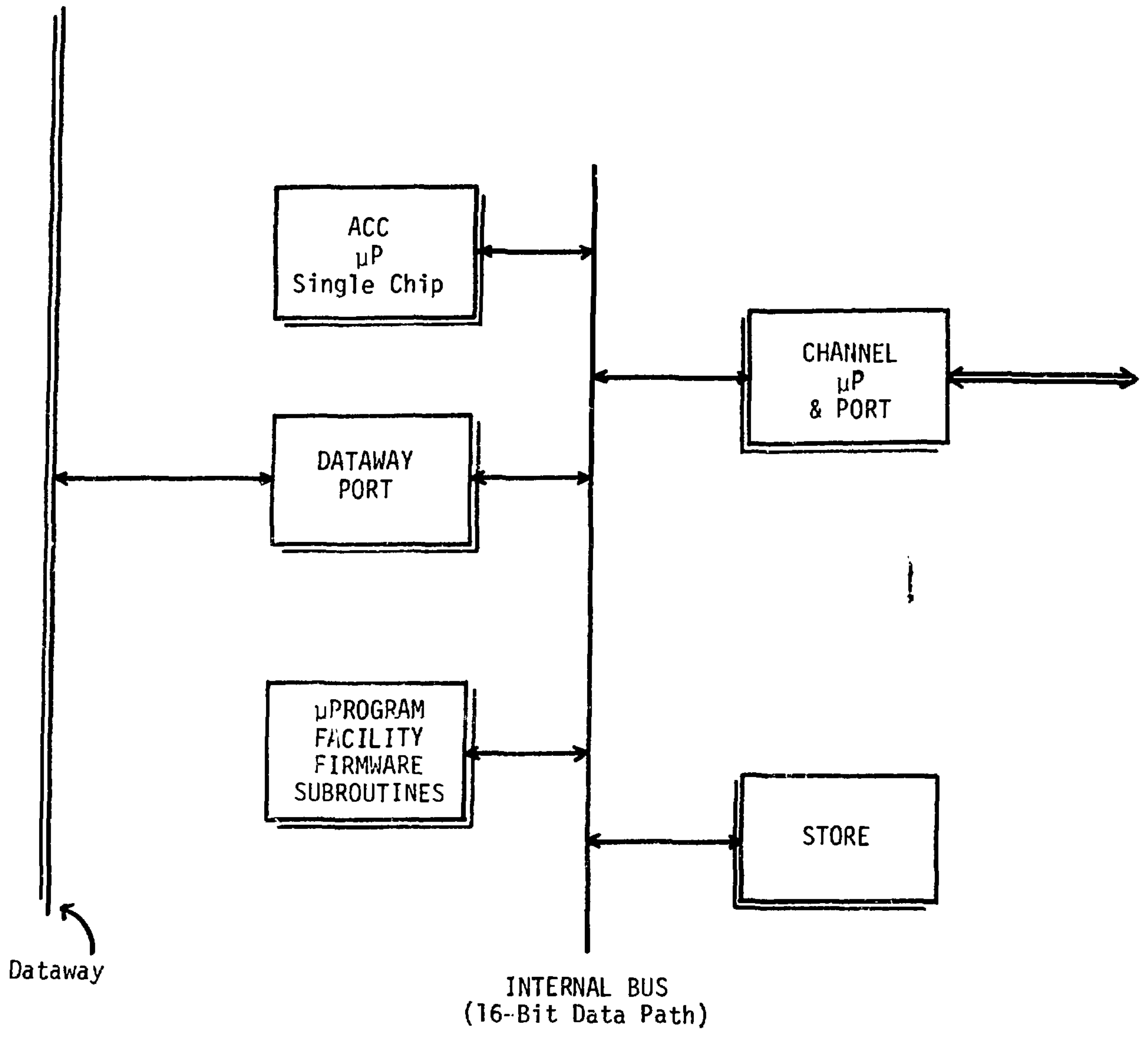


Fig. A-5. Recommended ACC configuration.

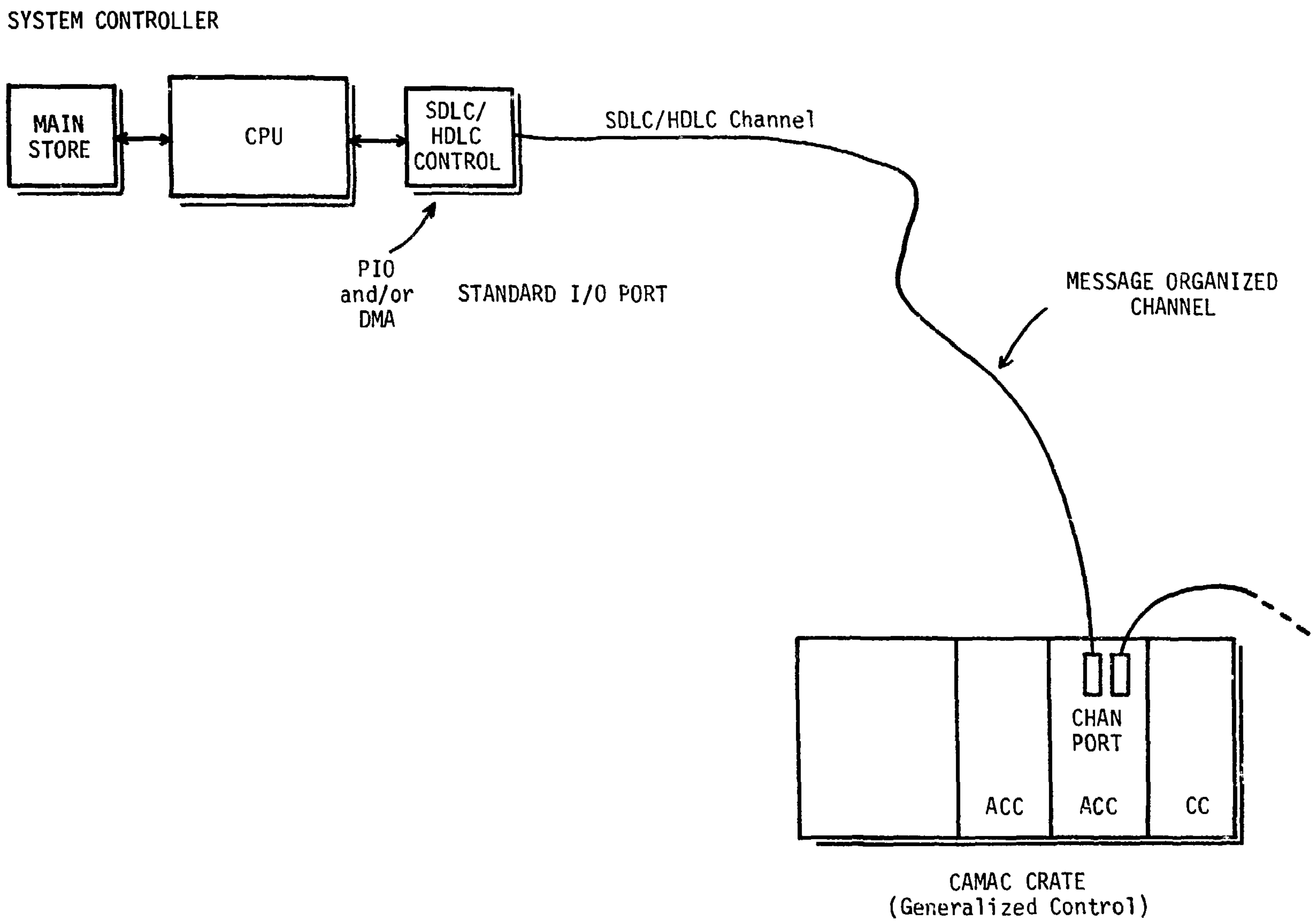


Fig. A-6. CAMAC communications channel.