



**Fermilab**

TM-1298  
0812.000

**CAMAC 488 Module  
68000 Based GPIB Interface Module**

K. C. Seino

March 1985

## LIST OF ILLUSTRATIONS

- Fig. 1 System Connections
- Fig. 2 C-Card Block Diagram
- Fig. 3 Timing Waveforms - Write Registers to FIFO
- Fig. 4 Timing Waveforms - FIFO to Read Registers
- Fig. 5 Timing Waveforms - CTM DMA
- Fig. 6 Timing Waveforms - MTC DMA
- Fig. 7 M-Card Block Diagram
- Fig. 8 Memory Map
- Fig. 9 Byte Manipulation
- Fig. 10 Front Panel
- Fig. 11 Software System
- Fig. 12 GPIB Test Page
- Fig. 13 Spectrum Analyzer Page
- Fig. 14 Spectrum Analyzer Display

Table 1 Interrupt Vector Assignment

1.0	INTRODUCTION . . . . .	1
1.1	Main Features . . . . .	2
2.0	HARDWARE . . . . .	2
2.1	C-Card . . . . .	3
2.1.1	Block Diagram . . . . .	3
2.1.2	CAMAC Interface . . . . .	3
2.1.3	FIFO Control . . . . .	4
2.1.3.1	CTM FIFO Control . . . . .	4
2.1.3.2	MTC FIFO Control . . . . .	5
2.1.4	DMA Control . . . . .	5
2.1.4.1	CTM DMA Control . . . . .	5
2.1.4.2	MTC DMA Control . . . . .	5
2.1.5	GPIB Interface . . . . .	6
2.1.6	TCK Decoder . . . . .	6
2.2	M-Card . . . . .	7
2.2.1	Block Diagram . . . . .	7
2.2.2	Memory Map . . . . .	7
2.2.3	DMA Controller . . . . .	8
2.2.4	Interrupt Controller . . . . .	8
2.2.5	Byte Manipulation . . . . .	9
2.3	Front Panel . . . . .	10
2.4	Hardware Tests . . . . .	11
2.4.1	Tests With Emulator . . . . .	11
2.4.2	Tests Under GBUG . . . . .	11
2.4.3	CAMAC Command Test . . . . .	11
2.4.4	TCK Test . . . . .	12
2.4.5	Overall Module Tests . . . . .	12
3.0	SOFTWARE . . . . .	13
3.1	GBUG . . . . .	13
3.2	GAS . . . . .	13
3.3	PROTO . . . . .	14
3.4	GAG . . . . .	15
3.4.1	System Environment . . . . .	15
3.4.2	Interior Design . . . . .	15
3.4.3	Data Flow . . . . .	15
3.5	Application Pages . . . . .	16
3.5.1	GPIB Test Page . . . . .	16
3.5.2	Parameter Page . . . . .	16
3.5.3	Spectrum Analyzer . . . . .	16
4.0	CONCLUSION . . . . .	17
5.0	ACKNOWLEDGMENT . . . . .	17
6.0	REFERENCES . . . . .	17

APPENDIX A           CAMAC INTERFACE

APPENDIX B           FPLA DESIGN DETAILS

## 1.0 INTRODUCTION

E. Malamud of Tev I listed approximately twelve GPIB devices on his 7/22/83 document. Now, how do we integrate such devices into the ACNET computer system (hardware and software)?

Setting up a device locally may be accomplished by pushing ten buttons on the front panel. Each button pushed corresponds to a command, which is equivalent to a ASCII string sent to the device over the GPIB interface. Therefore, setting up a device remotely is accomplished by sending the correct sequence of the ten corresponding ASCII strings. Some commands result in the return of data. The data returns in different forms, i.e., ASCII strings, binary words, binary bytes and so forth. The amount of data varies from several bytes to 2K bytes.

What kind of hardware and software should we come up with to interface GPIB devices with the existing computer system? One idea which D. Bogert suggested was to use a commercially available Multibus card, BLC 8488 from National Semiconductor, whose on-board Z80 would manage the GPIB read/write functions and handshakes. With this card, one could make a hardware system which would consist of (1) CAMAC OBO, (2) Multibus crate, (3) M. Shea's M68000, (4) MOBO, (5) BLC 8488 and (6) memory board. And the software considered for such a hardware package was GAS, which had been an established software package for communication between the ACNET computer system and smart CAMAC modules.

D. Beechy pursued the idea and put a system together for the bakeout system in a relatively short period of time. However, a few other people suggested another idea. The idea was to put everything on a two-wide CAMAC module. They made a comment like 'It is ugly and wasteful to have a two-wide CAMAC module, a Multibus crate and several cards in it just to interface to GPIB devices.

The author pursued the second idea and came up with a two-wide CAMAC module called C488. It was not easy to reduce the space. Needless to say, it was a completely new design. When the artwork was generated, we had to allow 8 mil line/8 mil space and two lines between IC pads. The modification on the software (GAS) turned out to be simple and it was completed in few days.

The author will describe the hardware - block diagrams, circuit blocks, front panel and hardware tests. He will also refer to the software - system, modules and applications. The software was done by L. Chapman, S. Morris and W. Marsh. If the reader wants to know more about the software, he should read the references listed in this report and/or talk to the programmers mentioned above.

## CAMAC 488 Module: V 1.0

## 1.1 Main Features

- \* Motorola MC68000, which is clocked at 8 MHz, is the CPU of the module.
- \* Hitachi HD68450, which is clocked at 8 MHz, is the DMA controller. The channel assignments are Ch.0 for CAMAC to M-bus data transfer, Ch.1 for M-bus to CAMAC data transfer and Ch.2 for M-bus from/ to GPIB.
- \* Two FIFOs are on the module, i.e., one for CAMAC to M-bus data transfer and the other for M-bus to CAMAC data transfer. The size of the FIFOs is 64 bytes each.
- \* Memory sizes are 16K bytes of GBUG PROM, 8K bytes of GAS PROM, 8K bytes of PROTO PROM and 64K bytes of RAM.
- \* AMD 9519A is the interrupt controller. The interrupt level assignments are Lvl.2 for HINT, Lvl.3 for EOP1, Lvl.4 for EOP2, Lvl.5 for GPIB and Lvl.6 for 15 Hz.
- \* TI TMS9914A provides an interface between the M-bus and the GPIB specified in IEEE-488 1975/ 78 standards and IEEE-488A 1980 supplement. The device is programmable and can function as a controller, a talker or a listener.
- \* The module has two RS232 ports, i.e., one for an optional local terminal and the other for an optional remote RS232 device.
- \* The module communicates with the ACNET computer system via a modified O80 to M68000 version of GAS.
- \* It is a two-wide CAMAC module. The M-Card can be easily detached from the rest by removing two screws and two ribbon cables.
- \* Connections to GPIB devices and a RS232 device are made via a patch panel.

## 2.0 HARDWARE

The GPIB module consists of C-Card, M-Card and a front panel. The front panel forms a module, physically fastening the two cards together. The electrical connections between the cards are made through a 50 conductor ribbon cable/ connector assembly.

The overall connections between the module and peripherals are shown in Fig. 1. By connecting a terminal to a RS232 port which is located on the front panel, one can do basic diagnoses on the module. Another RS232 port and a GPIB port are brought out to a patch panel from the I/O connectors of the cards. The patch panel can be mounted

on either the front or rear of the relay rack. From the patch panel, one can make connections to devices with GPIB and/ or a device with RS232 interface.

## 2.1 C-Card

The C-Card constitutes the right half of the module, and its prime function is to interface with CAMAC. The name 'C' came from CAMAC.

### 2.1.1 Block Diagram -

A block diagram of the C-Card is shown in Fig. 2. The data is transferred in two directions, i.e., from CAMAC to M-bus (CTM) and M-bus to CAMAC (MTC). In case of the CTM direction, a CAMAC command writes data into a set of registers, and the data is transferred from the registers to the CTM FIFO whenever the FIFO input is ready. The data then wait for a DMA operation. A DMA controller is armed by the software, and it starts an operation when it receives the first request. The DMA control circuit sends requests to the DMA controller, and the controller returns acknowledges back. When a predetermined number of data bytes have been transferred from the FIFO to the memory, the controller terminates the operation. The control circuit receives a DONE signal from the controller at the end of operation, and it is reset by the signal.

The DMA operation for the MTC direction is almost same as the CTM except the fact that data bytes are shifted into the MTC FIFO from the memory by DTC signals. A DTC signal is generated by the DMA controller toward the end of the acknowledge period. Whenever a set of registers are empty and whenever the MTC FIFO has some data, the data bytes are transferred from the FIFO to the registers. A CAMAC command reads them out of the registers.

### 2.1.2 CAMAC Interface -

The CAMAC interface consists of write registers, read buffers and registers, status buffers, command buffers and decoders, module number buffers, a LAM generator and read/ write managers.

CAMAC commands are listed in APPENDIX A.1. Module status bits are listed in APPENDIX A.2. Conditional Q responses are illustrated in APPENDIX A.4.

The CAMAC write managing circuit consists of two flip-flops (U3, Schematic ED35871, SH-1/3), two AND gates (U62) and a one-shot (U50). When the first write command clocks the flip-flop on the left

(referring to the schematic), it stays as reset with its D-input being initialized to be low. Thus at the S1 time, CWRCK becomes true and latches write data into the write registers. At the same time, the one-shot is triggered and its output sets the flip-flop on the right indicating a write is in progress. When the data bytes have been transferred from the write registers to the FIFO, WRHEN arrives and resets the flip-flop. The same things repeat for write commands which follow the first one. However, if another write command arrives before the data bytes for the previous one are transferred, two things happen -- (1) CWIP becomes true, and thus the Q response is not generated, and (2) CWRCK becomes false, and thus the new data bytes are not overwritten into the registers.

The CAMAC read managing circuit consists of two flip-flops (U65), a NAND gate (U66), a OR gate (U27), a one-shot (U18) and other small gates. When read data bytes are transferred from the FIFO to the read registers, the flip-flop on the right is set by RRHCK indicating that the read data is valid. When a read command arrives and clocks the flip-flop on the left, the flip-flop is set. In this situation, two things happen -- (1) CRDVAL becomes true and thus Q is generated, and (2) CRREN becomes true and thus the data in the read registers is enabled for a read operation. However, if a read command arrives before the data becomes valid, Q is not generated and the read data is not enabled either.

### 2.1.3 FIFO Control -

#### 2.1.3.1 CTM FIFO Control -

The control circuit provides timing pulses for transferring data from the write registers to the CTM FIFO. The timing pulses are shown in Fig. 3. As illustrated in the figure, the low byte of the write registers is first enabled by WRLEN and shifted into the FIFO by FYSI. The middle and high bytes are enabled by WRMEN and WRHEN respectively, and shifted into the FIFO by FYSI, immediately after the low byte. Basic timing pulses are generated by two one-shots (U57, Schematic ED 35871, SH-1/3), a shift register (U56), a FPLA (U55), and other gates. A write command normally consists of 3 bytes, and thus 3 shift pulses are needed for the data transfer. However, when the write command brings in a header, the WTF period is extended and the fourth pulse shifts a dummy (assurance) byte into the FIFO. A one-shot (U79) and its associated gates determine whether it is a three byte transfer or a four byte transfer and produce a proper timing pulse to clear the control circuit.

### 2.1.3.2 MTC FIFO Control -

The control circuit provides timing pulses for transferring data from the MTC FIFO to the read registers. The timing pulses are shown in Fig. 4. As illustrated in the figure, the first byte is latched into the low byte of the read registers by RRLCK. The second and third bytes are latched into the middle and high bytes of the read registers by RRMCK and RRHCK respectively. The leading edge of a clock pulse latches the data appearing at the output of the FIFO, and the lagging edge of the same pulse shifts out new data to the output of the FIFO.

### 2.1.4 DMA Control -

#### 2.1.4.1 CTM DMA Control -

CTM DMA control signals are shown in Fig. 5 for a header transfer. When a F2OAO command is properly received, HDRCK triggers a one-shot (U37, Schematic ED35871, SH-1/3), and after 1 microsecond delay, HSG becomes true at a flip-flop (U67). When FYOR1, 2 and HSG become true at U72, a one-shot (U61) is triggered and a flip-flop (U60) is set. The output of U60 becomes DMREGO when the software sets a flip-flop (U69) and when DMGOO1 becomes true. The software needs to arm the DMA controller chip first and then to allow the request to reach the chip in order to operate the chip in a cycle-steal mode. When the control circuit receives a DMACKO, it resets the request and generates the next request after a delay. It repeats this hand shaking four times for a header transfer. When the fourth DMACKO and DMDONE arrive, the DMREGO is reset and remains reset, and the HSG is reset at the lagging edge of the DMDONE. The DMGOO1 is reset by the software.

A data transfer following the header transfer is similar to the header transfer. It is started by a CTMCK, and it is terminated when a predetermined number of bytes (a multiple of three) have been transferred.

#### 2.1.4.2 MTC DMA Control -

MTC DMA control signals are shown in Fig. 6. When a header transfer has been properly performed for a MTC DMA operation, the software sets a flip-flop (U73, Schematic ED35871, SH-1/3) with MTCGO and MTC becomes true. Then a one-shot (U74) is triggered, and it sets a flip-flop (U73). When the software arms the DMA controller chip and makes DMGOO1 true, the output of the flip-flop is allowed to reach the DMA controller chip as a request. When the control circuit receives an acknowledge, it resets the request and sets the request back after

a delay following the lagging edge of the acknowledge. It is essential to hold back next request until the current data byte is shifted into the MTC FIFO at the end of the acknowledge period. The control circuit exchanges requests with acknowledges a predetermined number of times. At the last exchange, a DMDONE resets the MTC and DMREG1, and the request remains false. The software resets the DMGO01.

#### 2.1.5 GPIB Interface -

The interface circuit consists of a controller chip (U17, Schematic ED35871, SH-3/3), bus transceivers (U5, U16) and a OR gate (U27). The controller chip is the TMS9914A (Texas Instruments), which provides an interface between a microprocessor and the GPIB specified in the IEEE-488 1975/ 78 standards and the IEEE-488A 1980 supplement. The device is programmable and can function as a controller, a talker or a listener. For further details, one should refer to Ref. 1.

A device on the GPIB can request a service from the CPU via the GPIB controller chip. For example, a device has data ready and makes SRQ true on the GPIB. The controller chip sees it and interrupts the CPU. In an interrupt service routine, the CPU does a serial poll and reads the data.

Data can be transferred between the GPIB controller chip and the M-bus via DMA. When the software arms the Channel 2 of the DMA controller (U3, Schematic ED358871, SH-2/3) and makes the DMGO2 true, the GPIB controller chip and the DMA controller start to exchange DMREG2 with DMACK2 and start transferring data. When a predetermined number of bytes have been transferred, the DMA operation is terminated, and the software resets DMGO2.

#### 2.1.6 TCK Decoder -

A voltage comparator (U1, Schematic ED35871, SH-3/3) converts incoming signals to TTL signals. A one-shot (U3) and a decoder chip (U15) work together to extract 8 bit event codes from pulse trains. A FPLA (U4) detects up to eight event codes and produces a pulse when a particular code is detected. The GPIB module needs to be updated by a 15 Hz event. For this reason, the FPLA detects \$0F (15 Hz event code) and produces pulses, which are used as an interrupt to the CPU. A pulse generator (U26) works with a one-shot (U37) and a OR gate (U27) to provide back-up pulses when the TCK event pulses are not available. For further details, one should read Ref. 2.

## 2.2 M-Card

The M-Card constitutes the left half of the module, and it is a Motorola MC68000 based microcomputer. The name 'M' came from 'Motorola'. The card is called 'M-Card', and the bus on the card is called 'M-bus'.

### 2.2.1 Block Diagram -

A block diagram of the M-Card is shown in Fig. 7. The CPU is a Motorola MC68000, which is clocked by a 8 MHz clock. A Hitachi HD68450 controls DMA operations. The 68000 and 68450 are connected together hand in hand. All the signals that are needed for the bus come out from the two chips, and they are first tied together, then buffered and distributed to the bus. The PROM/ RAM area can accommodate up to seven pairs of memory chips, which can be either PROMs or RAMs, and which can be either 24 pinners or 28 pinners. Addresses are decoded by three FPLAs, which can be programmed for a particular application that one envisions.

The status/ control circuit consists of registers and a DIP switch. When the M-Card has a certain condition, its software sets a bit in a register to indicate the condition to the C-Card and the CAMAC. Or when the CAMAC wants to do a certain operation, it sets a bit on the C-Card. The M-Card reads it and knows what to do. Or an operator can set a bit on the DIP switch to indicate a certain operation to the M-Card.

A Signetics SCN68681 provides two RS232 ports. The first port is used with a local terminal for diagnoses. The second port is used for controlling a device with a RS232 interface or for downloading programs from the host.

### 2.2.2 Memory Map -

A typical memory map for the GPIB module is shown in Fig. 8. There are seven memory blocks, i.e., MEM0 thru MEM6, and these memory blocks are located on seven IC socket pairs.

MEM0 thru MEM3 are RAM blocks, each of which has 16K bytes. These RAM blocks are made reference to by the CPU in two address spaces, i.e., supervisor program or supervisor data except the first half of MEM0, which is accessible only in the supervisor data space.

MEM4 thru MEM6 are PROM blocks. MEM4 has 8K bytes, and a loader called PROTO resides in this block. MEM4 is located between 0 and 1FFF in the SP space and between 80000 and 81FFF in the SP and SD spaces. MEM5 has 8K bytes and accommodates GAS. MEM6 has 16K bytes and accommodates GBUG. These blocks are accessible in either SP or

SD.

The I/O block is accessible only in the SD space and it is located between FF8000 and FF9FFF. Base addresses for different devices are FF8000 (Status/ Control Bits), FF8101 (Interrupt Controller), FF8201 (GPIB Controller), FF8301 (Serial Ports) and FF8400 (DMA Controller).

For details on the memory mapping fuseware, one should read APPENDIX B.

### 2.2.3 DMA Controller -

The DMA controller is Hitachi HD68450, which has four independent DMA channels. Channel 0 is used for transferring data from CAMAC to M-bus, Channel 1 is for M-bus to CAMAC and Channel 2 is for GPIB from/ to M-bus.

The DMA controller (U3, Schematic ED35871, SH-2/3) is first armed by the software, and returns an acknowledge signal when it receives a request signal. The data is transferred during the acknowledge period. The controller generates a DONE signal after a predetermined number of data bytes have been transferred.

When we constructed a prototype module, we experienced difficulties in making a HD68450 work properly. We had to put pull-up resistors on some control signals of the chip.

When Channels 1 and 2 received requests simultaneously or very closely in time, the HD68450 could not properly sort them out and acknowledge them. If different priority levels were assigned on them, the chip should have been able to service two requests with two different priority levels. According to Hitachi, ones with R-mask had a deficiency in handling multiple requests. They said that they would replace ones with R-mask with ones with S-mask.

For further details on the HD68450, one should read Ref. 3.

### 2.2.4 Interrupt Controller -

The interrupt controller is AMD 9519A, which has Interrupt Request Register, Interrupt Service Register, Interrupt Mask Register, Auto Clear Register, Response Memory and others. Interrupt Register inputs are captured and latched in the Interrupt Request Register. Any requests not masked by the Interrupt Mask Register will cause a Group Interrupt output to the CPU. When the CPU is ready to handle the interrupt, it issues an Interrupt Acknowledge pulse, which causes (a) the priority of pending interrupts to be resolved and (b) a byte from the response memory (a vector number) associated with the highest

priority interrupt to be read.

The Group Interrupt output of the interrupt controller (U41, Schematic ED35871, SH-2/3) is connected to the Interrupt Control inputs (IPL<0:2>) of the CPU (U8). These connections allow us to use the Interrupt Level 6 of the CPU, which can be inhibited by the interrupt priority mask. Although we use only Level 6 of the CPU, we should remember that interrupt priority levels are assigned in the interrupt controller.

The CPU fetches a vector number from the interrupt controller, loads the program counter with the content of the interrupt vector and services the interrupt in an interrupt handling routine. Table 1 shows the interrupt vector assignment of the module.

For monitoring interrupt levels being serviced, the software turns on corresponding bits in a register (U66) and LEDs on the front panel.

For further details on the interrupt controller and interrupt handling, see Ref.'s 4 and 5.

#### 2.2.5 Byte Manipulation -

The GAS software package had been written for a Z80 based system, and hardware byte swapping and software four byte rotation were needed to make the GAS work on a 68000 based system. These manipulations are illustrated in Fig. 9.

When data words arrive with CAMAC commands F22 (or F16), as shown in Fig. 9b, the low byte is first shifted in the FIFO, and the middle byte and the high byte follow in order. On the 68000 based system, the high order byte has an even address that is the same as the word address and the low order byte has an odd address that is one count higher than the word address. In order to convert the data organization from Z80 to 68000, byte swapping is performed. The first byte out of the FIFO is normally stored at an even address by the upper data strobe (UDS). However, it is stored at odd address by the lower data strobe (LDS) instead. The second byte is stored at an even address by the UDS. These operations are accomplished by swapping UDS and LDS during the CAMAC to M-bus or M-bus to CAMAC DMA operations. The swapped UDS and LDS are called XUDS and XLDS respectively. Byte swapping is performed on all the pairs of bytes that follow the first.

When a header arrives with a CAMAC command F20, an assurance byte is added at the end after low, middle and high bytes have been shifted into the FIFO. When the header bytes are transferred from the FIFO to the memory, byte swapping is performed on the pairs of the bytes. After the header bytes have been stored in the memory, the software rotates them as illustrated in Fig. 9a. List Set command is shown in Fig. 9c as an example of byte manipulation.

Byte swapping on the M-bus to CAMAC data transfer is exactly the same as the one on the CTM transfer except the fact that it is performed in the reverse direction.

### 2.3 Front Panel

The front panel control/ monitor functions are shown in Fig. 10.

#### CAMAC

N: On when module is addressed.  
 LAM: On when LAM conditions exit.  
 SQ: On when module generates Q.  
 SX: On when module generates X.

#### Status

XTO: On when data transfer times out; possible data transfers are (1) Write Registers to CTM FIFO and (2) MTC FIFO to Read Registers.  
 EXH: On when header is expected.  
 HSQ: On when four bytes of header are being transferred.  
 CTM: On when CAMAC to M-bus data transfer is in progress.  
 MTC: On when M-bus to CAMAC data transfer is in progress.  
 FYIR: On when input of CTM FIFO is ready.  
 FZIR: On when input of MTC FIFO is ready.  
 FZOR: On when output of MTC FIFO is ready.

#### Control

ON: On when ON bit is true.  
 ENBL: On when ENBL bit is true.

#### M-bus

MPU HALT: On when 68000 is reset or stopped.  
 DMA<0:3>: On when DMA operations are in progress, numbers are associated with channel numbers.  
 INTR LEVEL<0:7>: On when interrupts are being serviced, numbers are associated with levels.  
 RESET: When this switch is pushed, it resets module hardware and reboots software.

#### GAS Managed Status

HB: Off when heart beat of module stops.  
 ICTI: On when I (GAS) can't take data in a set or list set command.  
 RUM: Off when module has unsolicited message for RLI.  
 HUM: Off when module has unsolidited message for host.  
 INI: Off when module just booted and waits for host to initialize it.

#### Power Supplies

+12V, +5V, -5V and -12V are monitored.

#### Misllaneous

LCL/ REM: By selecting LCL, operator can indicate to host that he wants to locally control GPIB devices.  
 TCK: On when TCK is detected.  
 TERMINAL: This is connector for RS232 terminal.

## 2.4 Hardware Tests

When the modules arrive from an assembly house, we first inspect them visually, and then measure resistivities between the ground and the power supplies. If all the above checks are good, we proceed to do the following tests.

### 2.4.1 Tests With Emulator -

If the GBUG does not work on the module, we can do trouble-shooting with an emulator. First, we try to write to the memory and read from the memory. Then, we try to read a few important locations of the PROTO and GBUG PROMs. Thirdly, we try to write to a register of the DUART (U49, Schematic ED35871, SH-2/3) and read from it. Lastly, we attempt to run the GBUG and break here and there along the way.

### 2.4.2 Tests Under GBUG -

With the GBUG working on the module, we can test out different ports. First, we write to a register and read from it on the UIC (U41, Schematic ED35871, SH-2/3) and the DMAC (U3). Secondly, we test control, status and interrupt level bits by writing to or reading from them. Thirdly, we test addressing to all the RAM and PROM pairs. Fourthly, we write to a register and read from it on the GBC (U17, Schematic ED35871, SH-3/3).

### 2.4.3 CAMAC Command Test -

Our CAMAC test facility consists of a CAMAC crate, Kinetic Systems' 5110 Multibus Adapter and 3908 Crate Controller and System 27 (Z80 based, with Multibus, CDOS 1.7 running). At this stage of the game, we test CAMAC commands F6A0, F9A0, F17A0, F26A0, F24A0, F30A0, F28A0 and F1A0. Then, to test SQ, we try FOA0, and to test SX, we try F10A0.

CAMAC 488 Module: V 1.0

#### 2.4.4 TCK Test -

We observe pulses at Pin 11 of U11 (Schematic ED35871, SH-3/3) to see if they are of 15 Hz. And by removing the TCK input to the module, we observe if the back-up clock circuit takes over. Lastly, we carefully look at the output waveform at pin 10 of U3 (one-shot).

#### 2.4.5 Overall Module Tests -

We developed module test programs on the EXORMACS development system. The programs are assembled, linked, built, downloaded to the module over the phone line and executed under the GBUG.

GBINT1 program was written to test out the interrupt handling. The 15 Hz clock pulses come in as interrupt requests, are acknowledged and turn on INTR LEVEL 6 LED in a service routine.

GBDMA4 program was written to test out CAMAC to M-bus and M-bus to CAMAC DMA operations. We first start running GBDMA4 under GBUG on the module, and we send or receive data from the CAMAC by running a CAMAC program. Two CAMAC programs, i.e., GBCTS1 and GBCTS2 were written in FORTRAN. With GBCTS1, we can test either the CTM data transfer or the MTC data transfer. We normally send a definite data pattern from the CAMAC to the M-bus and examine if the data pattern has been transferred without any error. We then transfer the same data pattern back from the M-bus to the CAMAC and examine it again. GBCTS2 does a CTM transfer first and does a MTC transfer without interruption of the program execution. It compares sent data with received data and counts the number of errors.

GB488D program was written to test the GPIB interface. The program initializes the bus, sends measurement parameters to Racal-Dana 6000 digital multimeter, sends a GET (Group Execute Trigger) and waits for a SRQ (Service Request). When the 6000 DMM has data ready, it sends a SRQ. The SRQ goes through the GPIB controller (U17, Schematic ED35871, SH-3/3) and the interrupt controller (U41, Schematic ED35871, SH-2/3), and come through as Interrupt Level 6 of U41. The service routine for the interrupt does a serial poll and sets up a DMA operation for transferring the data from the DMM to the M-bus. When the DMA is done, another interrupt (Interrupt Level 4) occurs. The service routine for this interrupt displays the DMM data on the terminal. After the service routine, the main program sends another GET to the DMM for the next cycle. This process repeats itself until the program execution is stopped.

### 3.0 SOFTWARE

The software system for the GPIB module roughly looks as shown in Fig. 11. GBUG, GAS and PROTO are PROM-resident programs, and reserve 16K bytes, 8K bytes and 8K bytes respectively for their use. GAG and its support modules (i. e., OPERA and drivers) are downloaded from the ACNET to the RAM on the module.

#### 3.1 GBUG

GBUG is a PROM resident monitor program derived from Motorola's VMEBUG. If the system is rebooted with SWSO off (down position, high true TTL level), it will run GBUG. Under GBUG, one can examine all the components of the microcomputer, run programs and do trouble-shooting on them.

#### 3.2 GAS

GAS (GHASP Advanced Software) is a PROM resident software package derived from GHASP. GHASP (General Host And Subsystem Protocol) is a language for communication between smart modules and the ACNET computer system.

The simplest way for a master to collect data from a module is to send the module an address (STANC), wait for the module to generate the answer and then read the answer. These features are supported in GAS by Set commands and Read commands.

Most data collection is repetitive, and therefore it is efficient to send a set of addresses once along with information about how often to update the answer and then read the answer repetitively. GAS supports three types of list commands, i. e., the List Setup (LS), the List Read (LR) and the List Delete (LD).

All GAS commands are transmitted as one or more CAMAC commands. Each GAS commands starts with a F20. The remaining bytes of the header (for Read, Set and List Set commands) are sent by two F22s. For Read and List Read, a number of F0 or F4 CAMAC commands are then sent. For Set and List Set, a number of F16 CAMAC commands are sent after the header.

When hardware or software problems exist in smart module, it may be impossible to successfully execute any of the five kinds of GAS commands explained above. For a simpler form of testing, GAS may be put in Regurgitation Mode. In this mode, the master sends three bytes to the module with a F20 command, GAS does the CTM and MTC DMA operations and simply returns the same three bytes. The master then read the three bytes with a F0 command.

In addition to supporting GAS commands, the GAS software controls some status bits. The master reads them with a F1 command. The status bits are as follows.

HB: 0 when heart beat of module stops.  
 ICTI: 1 when I (GAS) can't take data in set or list set command.  
 RUM: 0 when module has unsolicited message for RLI.  
 HUM: 0 when module has unsolicited message for host.  
 INI: 0 when module just booted and waits for host to initialize it.

GAS reports errors to the master by placing an error message in an unsolicited message queue. Whenever the host's queue is not empty, the module raises a LAM by clearing the HUM bit, requesting the host to read the queue. Whenever the RLI's queue is not empty, GAS clears the RUM bit but this does not cause a LAM. GAS reports system events as well as error messages to the masters via the unsolicited message queues. Each GAS error and system event may be bypassed. If bypassed, no message is generated even if the error or event occurs.

For further details on GAS, one should read Ref. 6.

### 3.3 PROTO

PROTO is a PROM resident program which knows how to download GAG. When the power is applied to the GPIB module, the CPU fetches the initial supervisor stack pointer and the initial program counter at Address Locations 0 and 4 of PROTO, it starts executing PROTO from the location pointed to by the initial program counter. When PROTO is ready for GAG to be downloaded, it flashes INTR LEVEL 7 LED. In order to download GAG, one must use a "Download Microp" application page. In the future, the ACNET will automatically detect the reboot of PROTO and will automatically download GAG.

If one wishes to run GBUG, he turns SWSO off (down position) before applying the power to the module. One of the things that PROTO does when it is just starting is to read the status on SWSO. If the switch is off, PROTO makes a jump from itself to GBUG. From this point on, everything is controlled and monitored by GBUG unless the system is restarted.

PROTO is assembled and linked assuming that the beginning of the program is located Address 80000 (HEX). However, since the memory mapping circuit allows it to be accessed at either Location 0 or Location 80000, the CPU can fetch initial stack pointer and program counter from Locations 0 and 4 without any problem at the time of system start.

### 3.4 GAG

#### 3.4.1 System Environment -

GAG is a software system which provides communications between ACNET and GPIB devices. GAG also provides translation between the ACNET language called GAS and whatever language the GPIB devices speak.

GAG allows three masters to exist, i.e., ACNET, a resident local master and an optional local terminal. The local master software task resides in the same microcomputer as GAG and can use GAG to communicate with the GPIB devices. This ability, together with the task's own intelligence, lets it do local control such as closed loops. The local terminal is intended for debugging and not a part of the final system.

In addition to GPIB devices, GAG has a port for connection to an optional RS232 device. When a software driver for this port is written and added to GAG, the RS232 appears to be just another GPIB device to the three masters.

#### 3.4.2 Interior Design -

GAG runs under a simple multi-task operating system called OPERA, which is non-preemptive and of a round-robin.

GAS has two communication modes: transparent and opaque. In the transparent mode, GPIB ASCII strings are sent between ACNET and GAG as data in GAS commands. In the opaque mode, the ASCII strings are stored in the GAG's translator which converts them to/from traditional data-base orientated GAS commands and data formats.

#### 3.4.3 Data Flow -

Each external device (RS232 or GPIB) has a FIFO queue associated with it. The three master tasks ask GAG to enter commands into the queues. The two external device driver tasks empty the queues, sending commands to a correct external device and returning status and data to the master who initiated the request.

The local resident master task can maintain a data pool of its own which is accessible to GAS.

The translation tables are downloaded to the microcomputer from ACNET via GAS and are used by GAG in the opaque mode.

For further details on GAG, one should read Ref. 7.

### 3.5 Application Pages

At the time of this writing, two application pages have been written. I am sure that there will be more developments in the future, i.e., changes and improvements will be made to the existing ones and new one will be added. The author will talk about three examples, which would give the reader some ideas how to communicate with remote GPIB devices from the console.

#### 3.5.1 GPIB Test Page -

A GPIB Test Page has been written as shown in Fig. 12. The operator first specifies a device name (the device has to be in the data base) and enters an ASCII string that he wants to send to the device. The device returns data (if any), which is displayed in HEX, ASCII or Integer.

In this crude way, the operator has to know the details of the device, i.e., how to operate the device and what specific ASCII string to perform a particular function.

#### 3.5.2 Parameter Page -

Some simple GPIB devices can be put on parameter pages. For example, the dipole magnet current measured by a DMM can be monitored on a parameter page. When the device name is entered and when the interrupt switch is pushed under the name, the software sends necessary commands to the device, brings data back and displays it on the page. It will be updated in a 1 Hz or 15 Hz rate. If the operator wants to control/monitor a few functions on the device, he can use the digital control/status facility. For example, if the operator wants to take a filter in and out of the DMM, he interrupts under FILTER IN OUT on the page. The information is carried on a bit of the GAS data, GAG translates it to a ASCII string, and the string is sent out to the DMM. The filter is thus manipulated on the DMM.

#### 3.5.3 Spectrum Analyzer -

A Spectrum Analyzer Remote Control and Display page was under development at the time of this writing. If one calls up the page, he sees a page like the one shown in Fig. 13. If one enters a data base name and terminates with an interrupt under INITIALIZE SPECTRUM ANALYZER MODULE =< >, the program is initiated and it draws a

analyzer display on the Lexidata like the one shown in Fig. 14. All the control/monitor functions including the CRT display of the spectrum analyzer are shown. When one interrupts under Update Spectrum Analyzer, the program is allowed to update the conditions of the analyzer and to enable the Lexidata cursor. By moving the cursor and interrupting under different functions, one can manipulate the spectrum analyzer from a console.

Furthermore, one can plot, accumulate and save the trace data, and he can plot the saved trace data.

#### 4.0 CONCLUSION

It was a big effort to squeeze the space from the one occupied by a two-wide CAMAC module and an Intel Multibus chassis to the one occupied only by a two-wide CAMAC module. Needless to say, it was a completely new design. When the artwork was generated, we had to allow 8 mil line/ 8 mil space and two lines between IC pads in order to have all the connections neatly organized and made short. The pc boards were nicely fabricated and assembled well with just a few shorts.

I had to use new chips to do an efficient and space saving design. The software staff was objectionable to this because they had to become familiar with new chips and to modify the hardware dependent module of the GAS. However, it turned out to be a relatively simple job. They simply copied parts of my hardware test programs.

The cost of the Multibus based system seems to be over \$3,000, which can be compared with \$1,500 for the new design. Furthermore, the new design saves real estate as a whole.

#### 5.0 ACKNOWLEDGMENT

I am grateful to Richard Klecka for his efforts on the GPIB module project. He constructed the prototype unit and did mechanical design for the production. He was involved in correcting mistakes on the artwork, inspecting pc boards and supervising pc board assembly.

#### 6.0 REFERENCES

1. TMS9914A General Purpose Interface Bus (GPIB) Controller Data Manual, Texas Instruments 1982.
2. Accelerator Controls Tevatron Time Clock System Clock Decoder, D. G. Beechy, June 1982.
3. HD68450 DMAC (Direct Memory Access Controller), Hitachi #U102.
4. Am9519A Application Note, AMD AMPUB-071.

5. 16-Bit Microprocessor User's Manual Third Edition, Motorola 1982.
6. Speaking GAS, ACNET Design Note No. 23.3, Lee J. Chapman, September 21 1983.
7. GAG Software (GAS to GPIB translator), ACNET Design Note No. 48.1, Lee J. Chapman, 15 August 1984.
8. Accelerator Controls CAMAC 488 - GPIB Controller, Drawing No. 0812-ED-35871, May 1984.

APPENDIX A  
CAMAC INTERFACE

A.1 CAMAC Commands

All commands return X. Return of Q is conditional for some commands.

F0A0	Read Read Registers, non-block transfer, Q is conditional.
F1A0	Read module status, always Q.
F4A0	Read Read Registers, block transfer reads, Q is conditional.
F6A0	Read module number, 0001EB (HEX), 488 (Decimal), always Q.
F9A0	Clear module, always Q.
F16A0	Write data into Write Registers, Q is conditional.
F17A0	Clear module and reboot, always Q.
F20A0	Write header into Write Registers, Q is conditional.
F22A0	Write additional header information to Write Registers, Q is conditional.
F24A0	Reset ENBL bit, always Q.
F26A0	Set ENBL bit, always Q.
F28A0	Reset ON bit, always Q.
F30A0	Set ON bit, always Q.

A.2 Status

F1A0 reads the following status bits.

R24:	XTO (Transaction Time Out) This bit is true (=1) if the following transaction times out: (1) Write Registers to CTM FIFO transfer and (2) MTC FIFO to Read Registers transfer.
R23:	EXH (Expecting Header) This bit is set true by F9A0, F17A0 or power-up. It is reset by proper receipt of F20A0.
R22:	HSQ (Header Sequence)
R21:	CTM (CAMAC to M-bus Sequence)

CAMAC INTERFACE

- R20: MTC (M-bus to CAMAC Sequence)
- R19: FYIR (FIFO-Y Input Ready)  
This bit true indicates that FIFO-Y is ready to accept data for CAMAC to M-bus transfer.
- R18: FZIR (FIFO-Z Input Ready)  
This bit true indicates that FIFO-Z is ready to accept data for M-bus to CAMAC transfer.
- R17: FZOR (FIFO-Z Output Ready)  
This bit true indicates that FIFO-Z has data ready CAMAC read.
- R16, R15 and R14 are not used.
- R13: HB (Heart Beat)  
This bit true indicates that MPU is alive and updating status on M-bus.
- R12: ICTI (I Can't Take It)  
This bit true indicates that GAS can't take data in set or list set command.
- R11: RUM (RLI Unsolicited Message)  
This bit false indicates that this module has unsolicited message for RLI.
- R10: HUM (Host Unsolicited Message)  
This bit false indicates that this module has unsolicited message for host.
- R9: INI (Initialize)  
This bit false indicates that this module just rebooted and that it needs to be initialized.
- R8: Not used.
- R7: ON  
This bit true indicates ON mode.
- R6: ENBL (Enable)  
This bit true indicates ENABLE mode.
- R5, R4, R3, R2 and R1 are not used.

A.3 LAM Generation

LAM is generated by some of the status bits, i.e., HB, HUM and INI.

A.4 Conditional Q Responses

Q is generated by ANDing the following conditions for a given command.

	F0	F4	F16	F20	F22	
CRDVAL	1	1	0	0	0	CAMAC Read Data Valid
CWIP			0	0	0	CAMAC Write In Progress
EXH				1		
FTR	0	0				FIFO-Z to RR transfer
FZOR			0	0	0	
HSQ	0	0		0		

## APPENDIX B

## FPLA DESIGN DETAILS

## FPLA01

```

/WRHEN = /(TA3*/TA4*/TA5*WTF)
/WRMEN = /(TA2*/TA3*/TA4*/TA5*WTF)
/WRLN  = /(TA1*/TA2*/TA3*/TA4*/TA5*WTF)
FYSI   = DLYA2*WTF*[(TA1*/TA2*/TA3*/TA4*/TA5)
                  + (TA2*/TA3*/TA4*/TA5)
                  + (TA3*/TA4*/TA5)
                  + (TA4*/TA5*HSQ)]
RHSQ   = /RCTM*HSQ

```

## FPLA02

```

RRHCK  = TB3*/TB4*FTR*DLYB2
RRMCK  = TB2*/TB3*/TB4*FTR*DLYB2
RRLCK  = TB1*/TB2*/TB3*/TB4*FTR*DLYB2
FZSD   = DLYB2*FTR*[(TB3*/TB4*)
                  + (TB2*/TB3*/TB4)
                  + (TB1*/TB2*/TB3*/TB4)]

```

## FPLA03

```

/HDRCK = /(F1*/F2*F4*/F8*F16*SN*/FZOR*S1*/HSQ*/CWIP*EXH)
/CTMCK = /{[(F1*/F2*/F4*/F8*F16)+(F1*F2*F4*/F8*F16)]
          *SN*/FZOR*S1*/CWIP}
MTCK   = [(F1*/F2*/F4*/F8*/F16)+(F1*/F2*F4*/F8*/F16)]
          *SN*S1*/FTR*/HSQ*/CRDVAL
/SPCQ  = /(F1*/F2*F4*/F8*F16*SN*/FZOR*S1*/HSQ*/CWIP*EXH
          +[(F1*/F2*/F4*/F8*F16)+(F1*F2*F4*/F8*F16)]*SN*/FZOR*/CWIP
          +[(F1*/F2*/F4*/F8*/F16)+(F1*/F2*F4*/F8*/F16)]*/FTR*/HSQ*CRD

```

## FPLA04

```

/F30AOS1 = /(F1*F2*F4*F8*F16*SN*S1)
/F28AOS1 = /F1*/F2*F4*F8*F16*SN*S1

```

FPLA DESIGN DETAILS

$$\begin{aligned}
 /F26A0S1 &= /(/F1*F2*/F4*F8*F16*SN*S1) \\
 F24A0S1 &= /F1*/F2*/F4*F8*F16*SN*S1 \\
 /ZS2 &= /(Z*S2) \\
 F(16, 20, 22) &= \\
 & \quad [(/F1*/F2*/F4*/F8*F16)+(/F1*/F2*F4*/F8*F16) \\
 & \quad \quad +(/F1*F2*F4*/F8*F16)]*SN \\
 /CAMRE &= /{[(/F1*/F2*/F4*/F8*/F16)+(F1*/F2*/F4*/F8*/F16) \\
 & \quad \quad +(/F1*/F2*F4*/F8*/F16)]*SN}
 \end{aligned}$$

FPLA05

$$\begin{aligned}
 /BX &= /{[(/F1*F2*F4*F8*F16)+(/F1*/F2*F4*F8*F16) \\
 & \quad +(/F1*F2*/F4*F8*F16)+(/F1*/F2*/F4*F8*F16) \\
 & \quad +(/F1*F2*F4*/F8*F16)+(/F1*/F2*F4*/F8*F16) \\
 & \quad +(/F1*/F2*/F4*/F8*F16)+(F1*/F2*/F4*F8*/F16) \\
 & \quad +(/F1*/F2*/F4*/F8*F16)+(/F1*F2*F4*/F8*/F16) \\
 & \quad +(/F1*/F2*F4*/F8*/F16)+(F1*/F2*/F4*/F8*/F16) \\
 & \quad +(/F1*/F2*/F4*/F8*/F16)]*SN} \\
 /PRTQ &= /{[(/F1*F2*F4*F8*F16)+(/F1*/F2*F4*F8*F16) \\
 & \quad +(/F1*F2*/F4*F8*F16)+(/F1*/F2*/F4*/F8*/F16) \\
 & \quad +(/F1*F2*/F4*F8*/F16)+(F1*/F2*/F4*/F8*F16) \\
 & \quad +(/F1*F2*F4*/F8*/F16)+(F1*/F2*/F4*/F8*/F16)]*SN} \\
 /F9A0S1 &= /F1*/F2*/F4*F8*/F16*SN*S1 \\
 F17A0S1 &= F1*/F2*/F4*/F8*F16*SN*S1 \\
 /F6A0 &= /(/F1*F2*F4*/F8*/F16*SN) \\
 /F1A0 &= /F1*/F2*/F4*/F8*/F16*SN \\
 F(0, 4) &= /{[(/F1*/F2*F4*/F8*/F16)+(/F1*/F2*/F4*/F8*/F16)]*SN} \\
 NS1 &= SN*S1
 \end{aligned}$$

FPLA06

$$/CKEV1 = /B0*B1*B2*B3*/B4*/B5*/B6*/B7*DVAL)$$

FPLA12

$$\begin{aligned}
 /MEM6 &= /{[(/A12*A13*/A14*A15*A16*A17*A18*A19) \\
 & \quad +(/A12*A13*/A14*A15*A16*A17*A18*A19) \\
 & \quad +(/A12*/A13*A14*A15*A16*A17*A18*A19) \\
 & \quad +(/A12*/A13*A14*A15*A16*A17*A18*A19)] \\
 & \quad *[(/FC0*FC1*FC2)+(FC0*/FC1*FC2)]*AS} \\
 /MEM5 &= /{[A13*A14*A15*A16*A17*A18*A19] \\
 & \quad *[(/FC0*FC1*FC2)+(FC0*/FC1*FC2)]*AS} \\
 /MEM4 &= /{[(/A13*/A14*/A15*/A16*/A17*/A18*/A19) \\
 & \quad *[(/FC0*FC1*FC2) \\
 & \quad \quad +(/A13*/A14*/A15*/A16*/A17*/A18*/A19) \\
 & \quad \quad *[(/FC0*FC1*FC2+FC0*/FC1*FC2)]*AS]} \\
 /INTACK &= /FC0*FC1*FC2*AS)
 \end{aligned}$$

FPLA13

$$\begin{aligned}
 /MEM3 &= /[(A14*A15*/A16*/A17*/A18*/A19) \\
 & \quad *[(/FC0*FC1*FC2+FC0*/FC1*FC2)*AS] \\
 /MEM2 &= /[(/A14*A15*/A16*/A17*/A18*/A19)
 \end{aligned}$$

```

          *(/FC0*FC1*FC2+FC0*/FC1*/FC2)*AS]
/MEM1    = /[(A14*/A15*/A16*/A17*/A18*/A19)
          *(/FC0*FC1*FC2+FC0*/FC1*FC2)*AS]
/MEMO    = /{[(/A14*/A15*/A16*/A17*/A18*/A19)
          *(FC0*/FC1*FC2)
          +(A12*A13*/A14*/A15*/A16*/A17*/A18*/A19
          +/A12*A13*/A14*/A15*/A16*/A17*/A18*/A19)
          *(/FC0*FC1*FC2)]*AS}

```

## FPLA14

```

/SLDMAC  = /(/A8*/A9*A10*/A12*/A13*/A14*/FC1*FB000)
/SLSVRT  = /(A8*A9*/A10*/A12*/A13*/A14*/FC1*FB000)
/EXOP    = /[(/A8*/A9*A10*/A12*/A13*/A14
          +A8*A9*/A10*/A12*/A13*/A14)*FC1*FB000]
/SLBGC   = /(/A8*A9*/A10*/A12*/A13*/A14*/FC1*FB000)
/SLUIC   = /(A8*/A9*/A10*/A12*/A13*/A14*/FC1*FB000)
/WMBSC   = /(/A1*/A8*/A9*/A10*/A12*/A13*/A14*/FC1*FB000*/ROW)
/RMBSC   = /(/A1*/A8*/A9*/A10*/A12*/A13*/A14*/FC1*FB000*ROW)
/MBI     = /(A1*/A8*/A9*/A10*/A12*/A13*/A14*/FC1*FB000)

```

**Fig.1 GPIB Module - System connections**

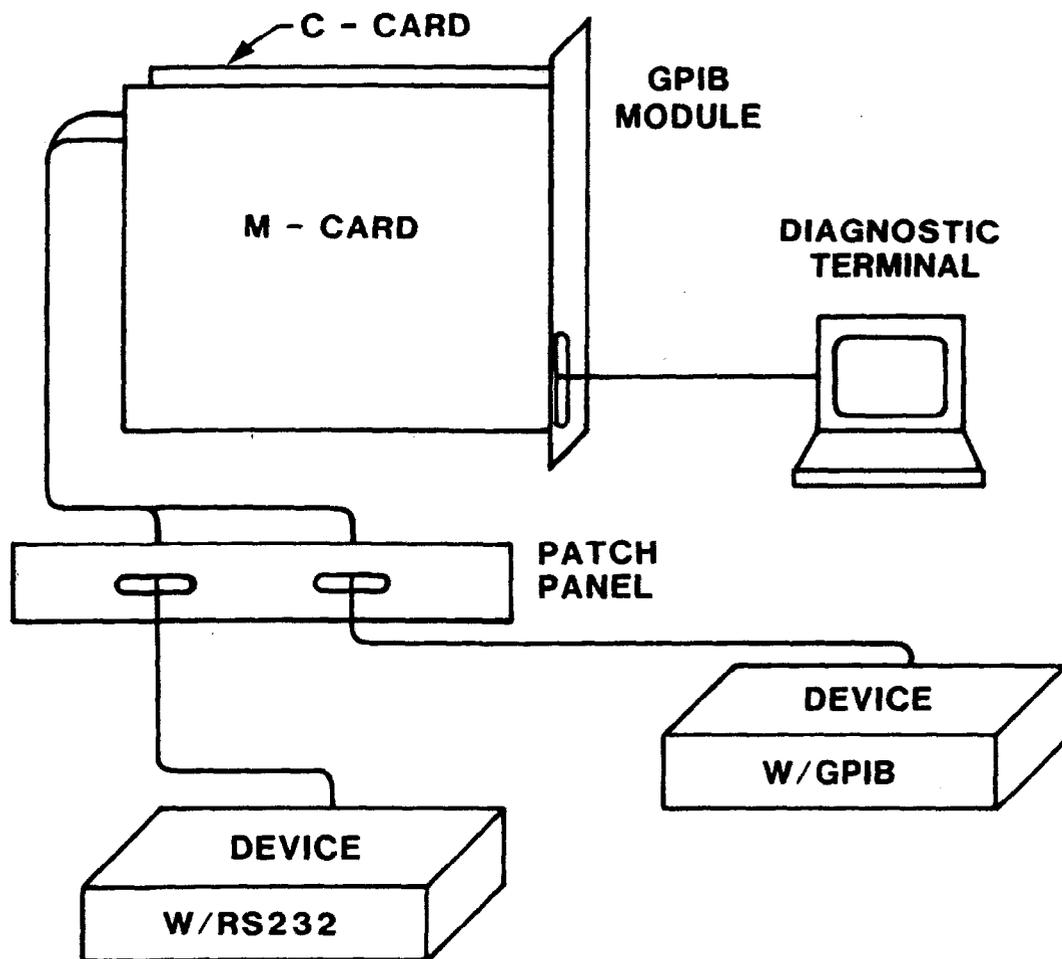
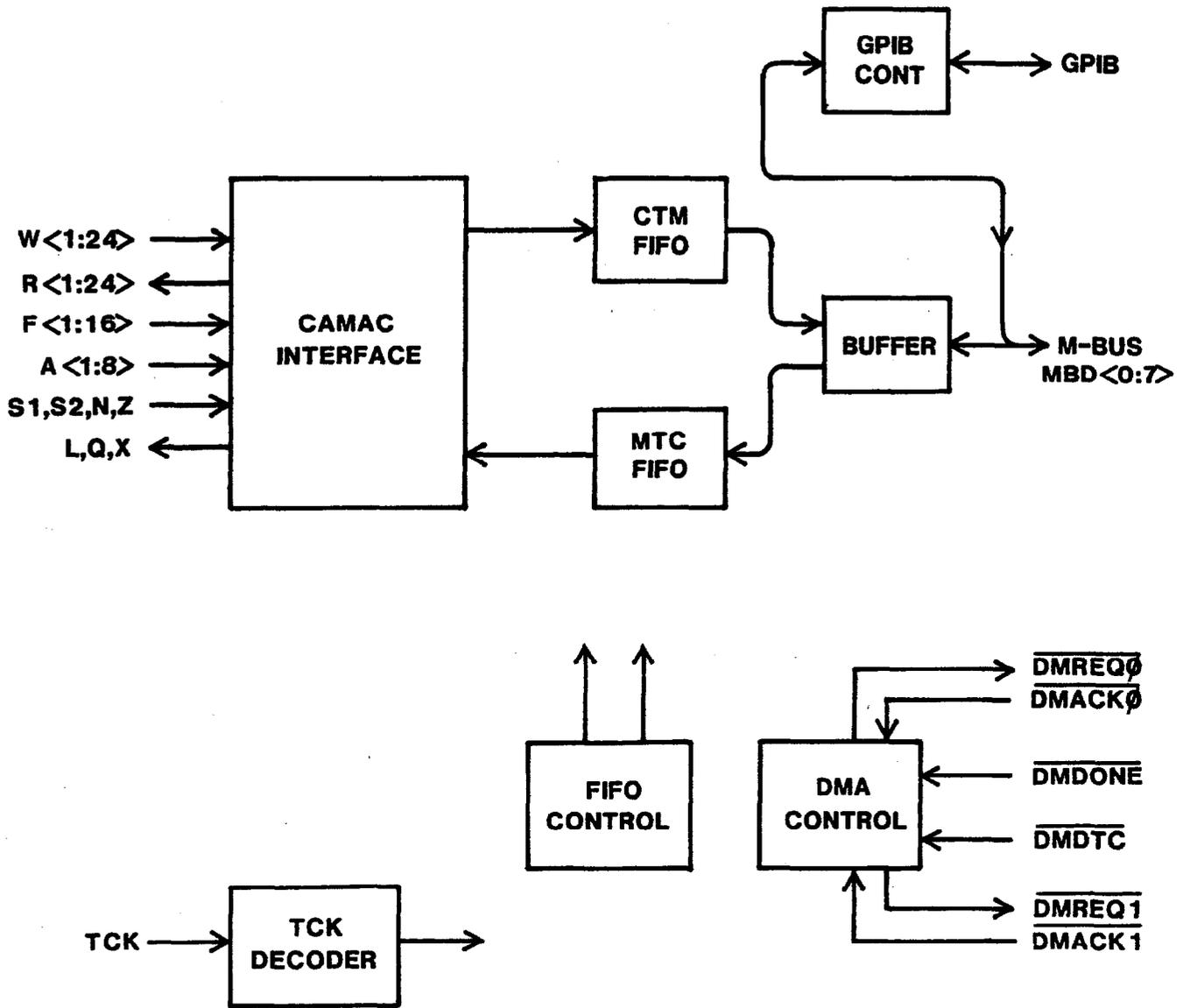
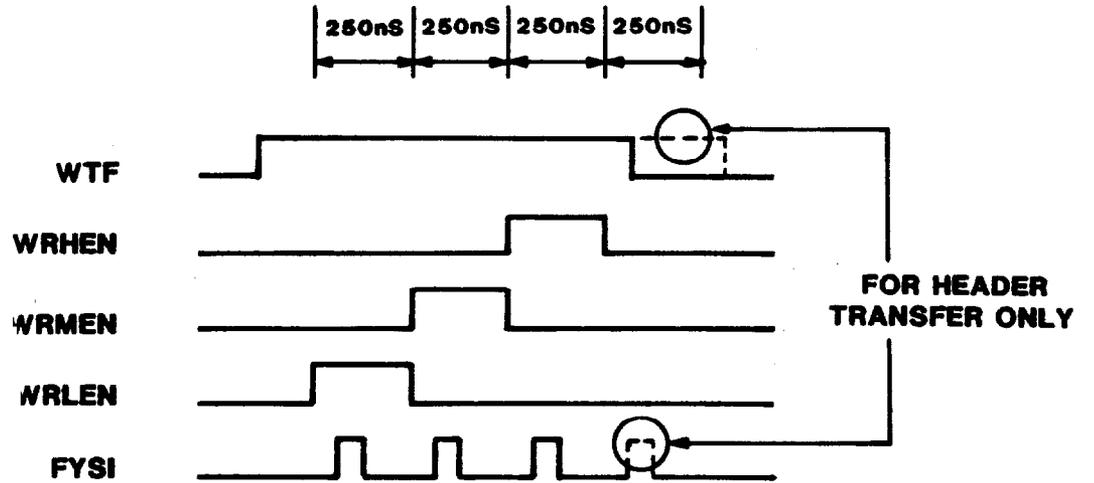


Fig.2 GPIB Module - C-Card block diagram



**Fig.3 Timing waveforms - Write registers to FIFO**



**Fig.4 Timing waveforms - FIFO to read registers**

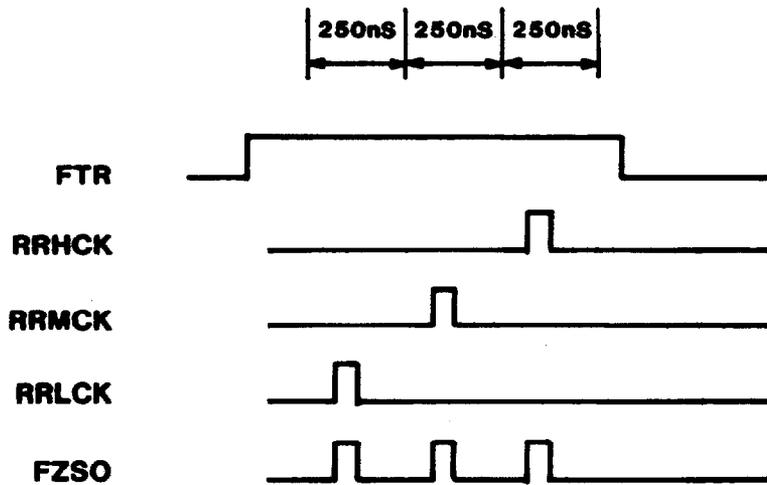


Fig.5 Timing waveform - CTM DMA

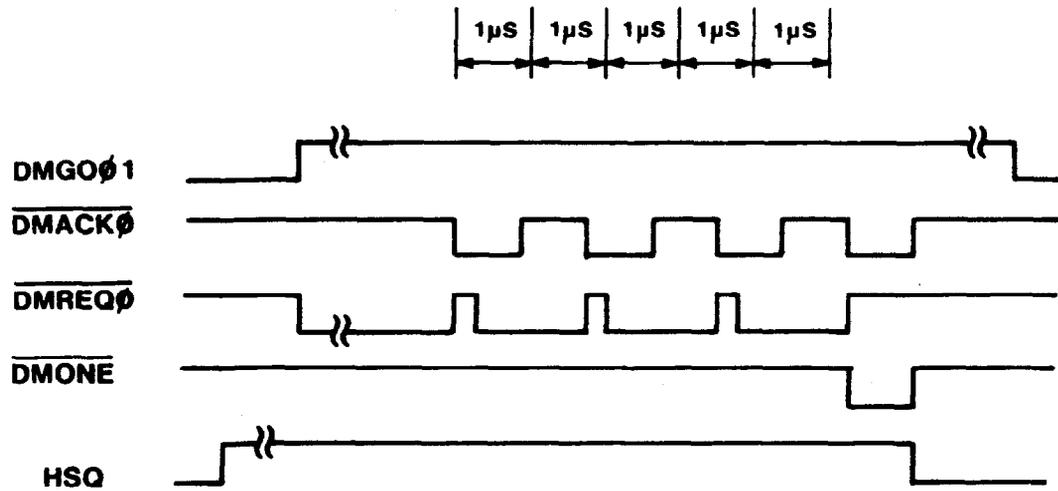


Fig.6 Timing waveform - MTC DMA

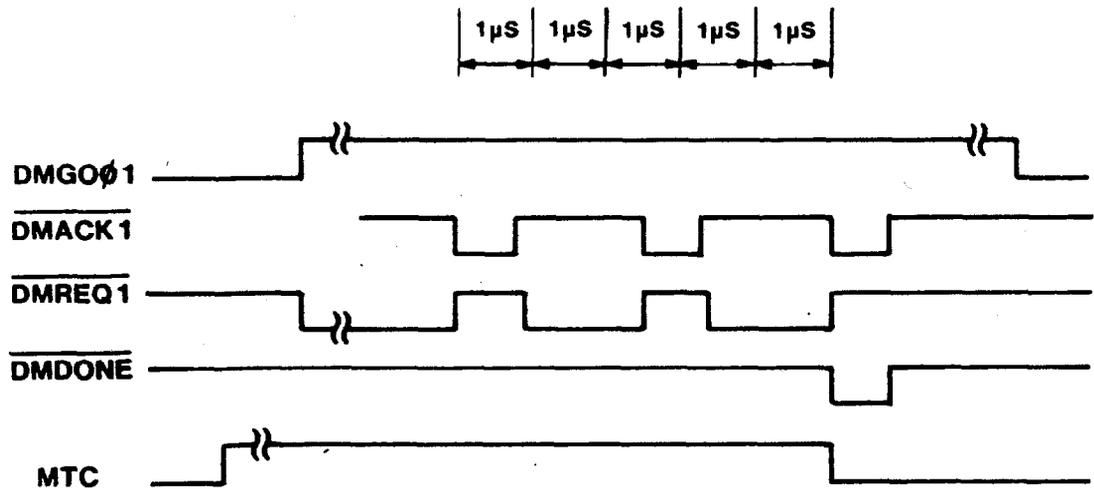
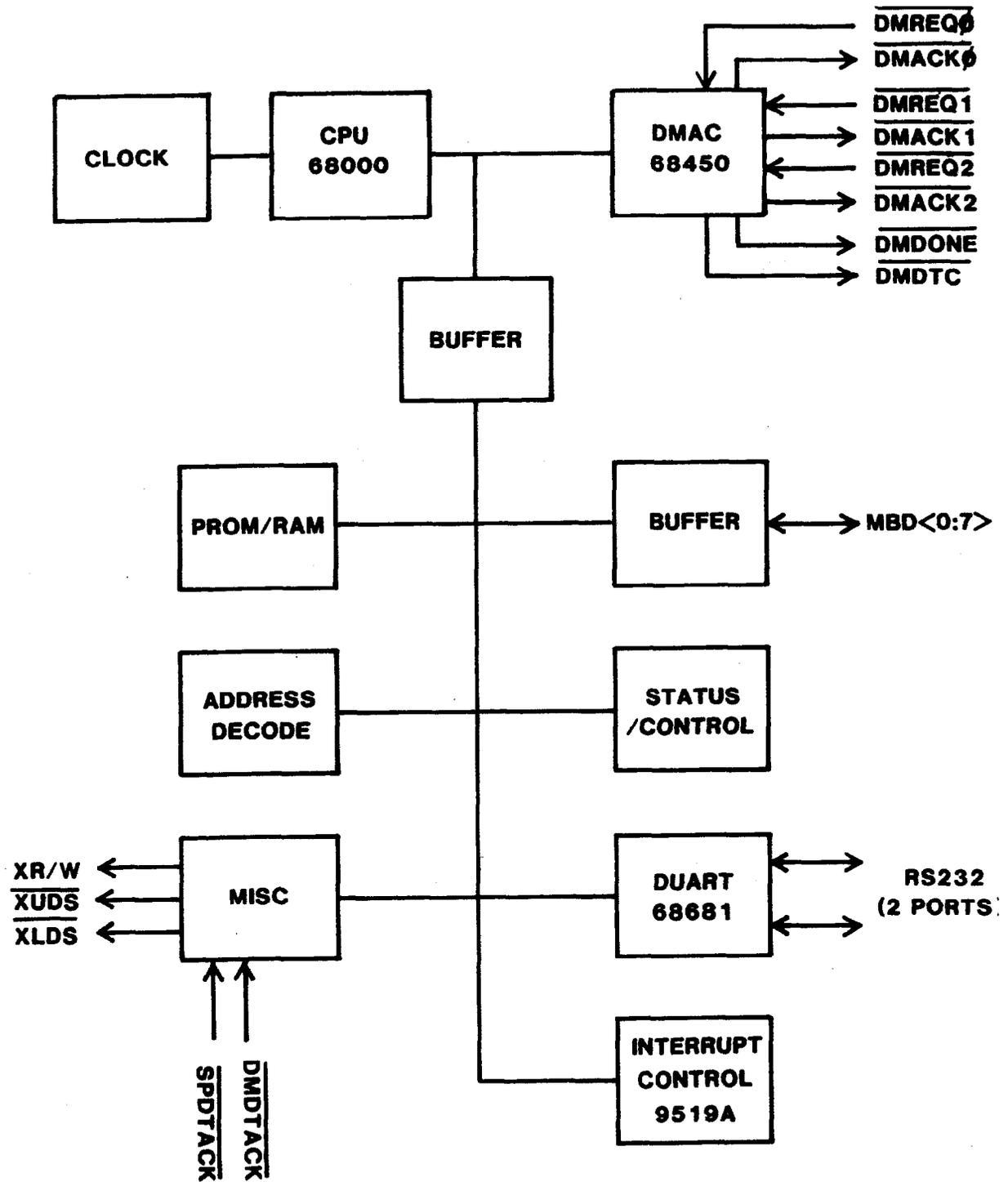


Fig.7 GPIB Module - M-Card block diagram



**Fig.8 GPIB Module - Memory map**

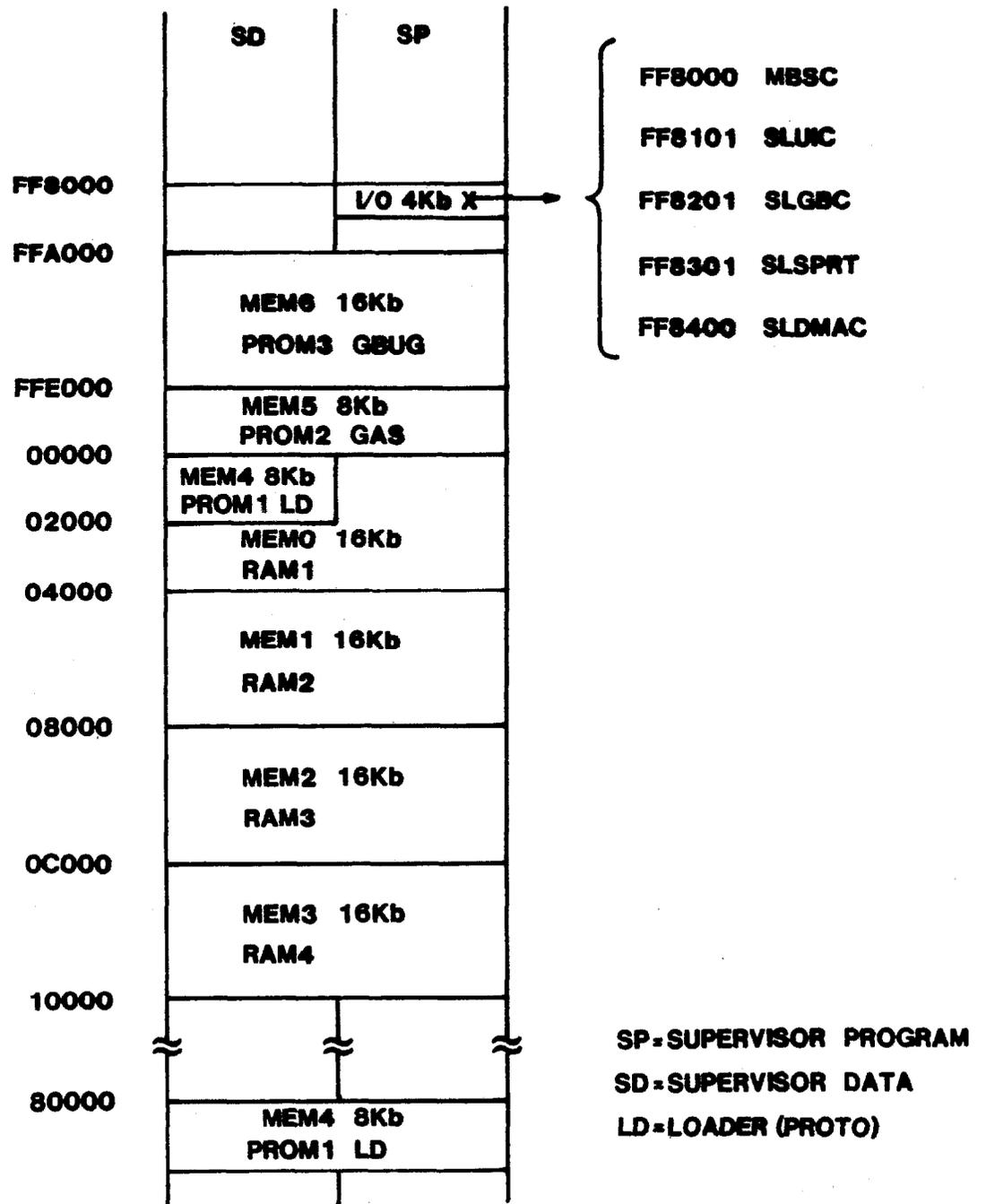


Fig. 9 GPIB Module - Byte Manipulation

(a) Header

:03:04:01:	F20 Data Word
:01:04:03:AS:	Shifted in FIFO
:04:01:AS:03:	Transferred from FIFO to memory
:01:AS:03:04:	Rotated in memory

(b) Data

:21:10:11:	F22 (or F16) Data Word 1
:30:31:20:	" Data Word 2
:51:40:41:	" Data Word 3
:60:61:50:	" Data Word 4
:11:10:21:20:31:30:41:40:51:50:61:60:	In FIFO
:10:11:20:21:30:31:40:41:50:51:60:61:	In Memory

(c) LS (List Set)

:LBCH:LBCL: TC :	Header
: NS : ANF: LID:	Data Word 1
:FTDH:FTDL: XX :	Data Word 2
: T : BCH: BCL:	Data Word 3
: NH : NL : A :	Data Word 4
: XX : SCH: CL :	Data Word 5

In Memory

: TC : AS :LBCH:LBCL: ANF: LID: XX : NS :FTDH:FTDL: BCH: BCL:
: A : T : NH : NL : SCH: CL : XX :

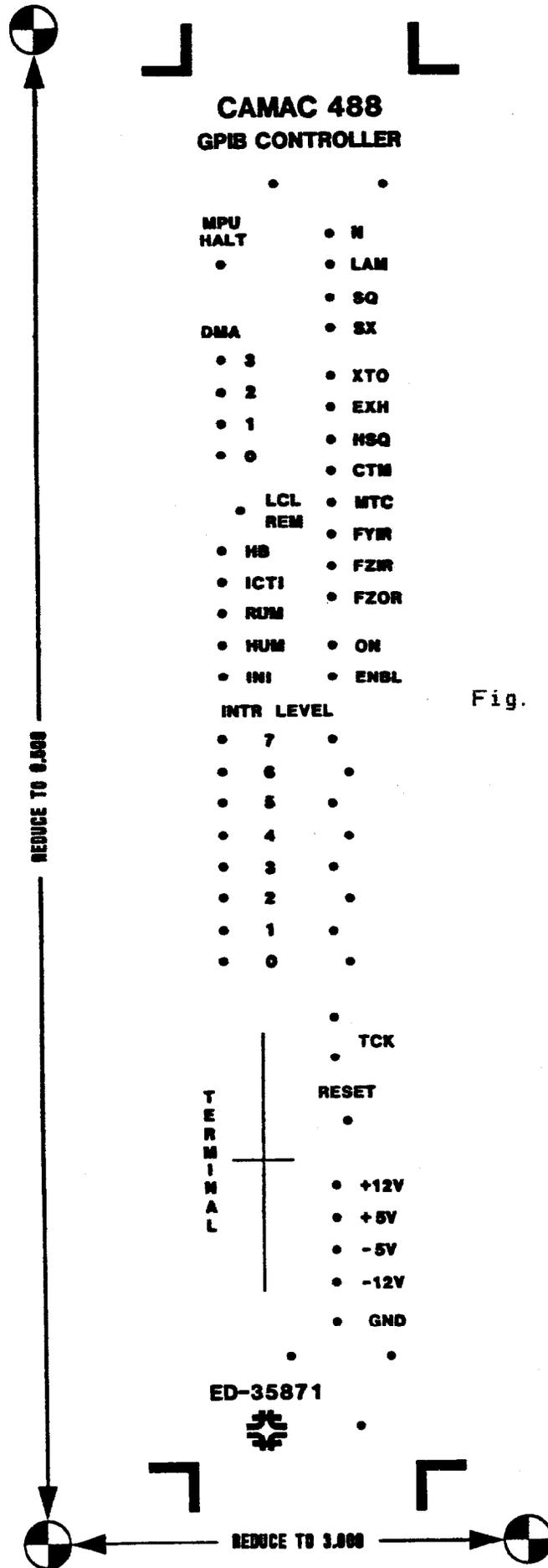
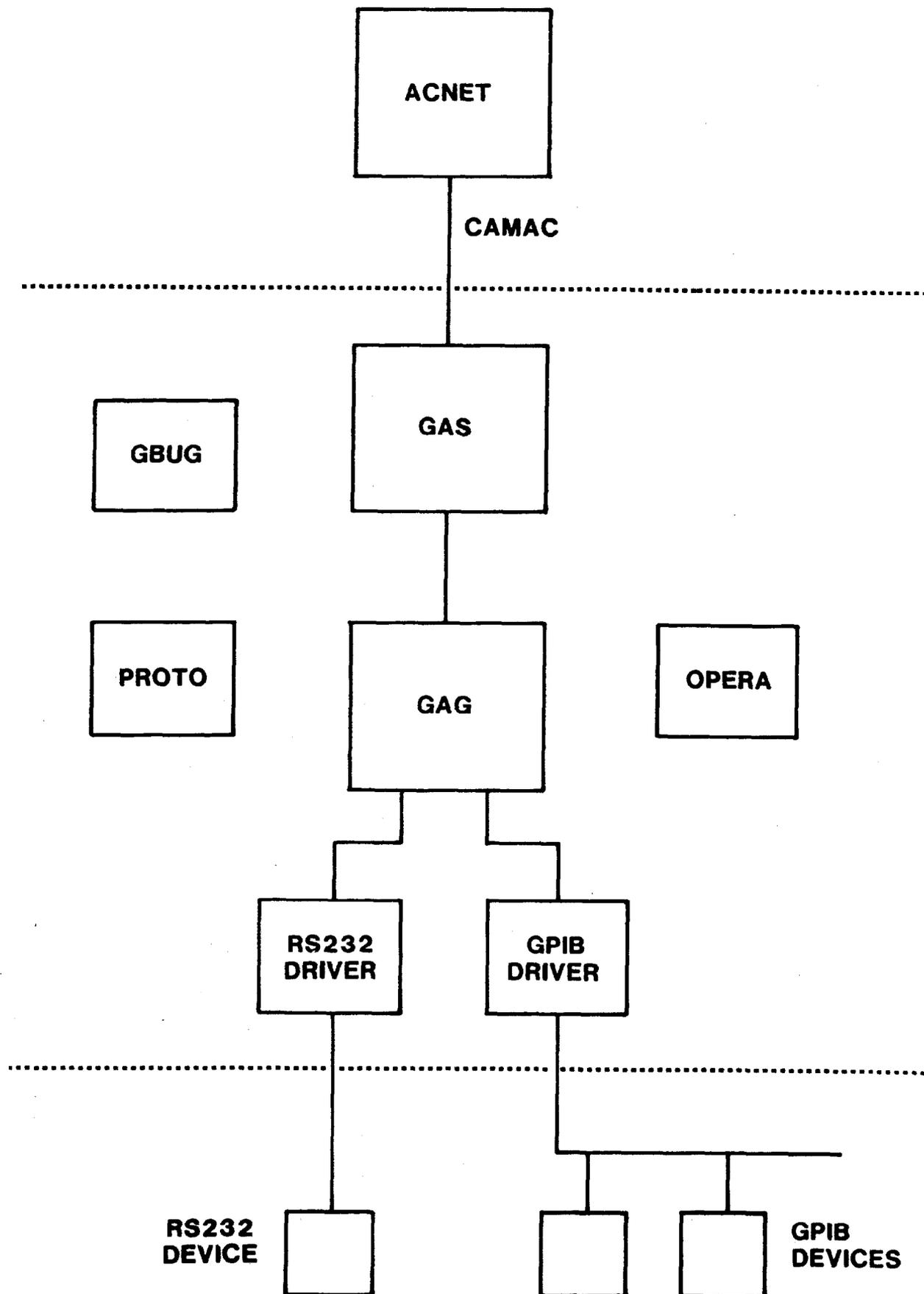


Fig. 10 GPIB Module  
- Front Panel

Fig.11 GPIB Module - Software system





WED 13-FEB-85 15:02

Z6 SPECTRUM ANALYZER REMOTE CONTROL AND DISPLAY TVTOSS ###

■ INITIALIZE SPECTRUM ANALYZER MODULE = <Z:1TSAT >

COMMAND FUNCTIONS

```
-----  
| #Update Spectrum Analyzer      |  
| #Enable LX cursor              |  
| #                               |  
| #CRT Mode #ON / #OFF          | #Help -< 1>+ #Edit Help Pages  
|-----
```

DATA ACCUMULATION/PLOTTING

```
-----  
| #Accumulate Trace Data        | #Plot trace data #Fill Mode  
|-----
```

SAVE/RESTORE

```
-----  
| #File Directory #Plot Trace File < > #Save Trace #Fill Mode  
|-----
```

ANALYZER COMMUNICATIONS / ERRORS

```
-----  
| 11 AFE = .FALSE.  
| 12 LEXCUR = .FALSE.  
| 13 LNDB = .FALSE  
| 14 split = .FALSE  
| 15 EPLOTT = .FALSE  
| 16 DISFIL = .FALSE  
|-----
```

Fig. 13 Spectrum Analyzer Page

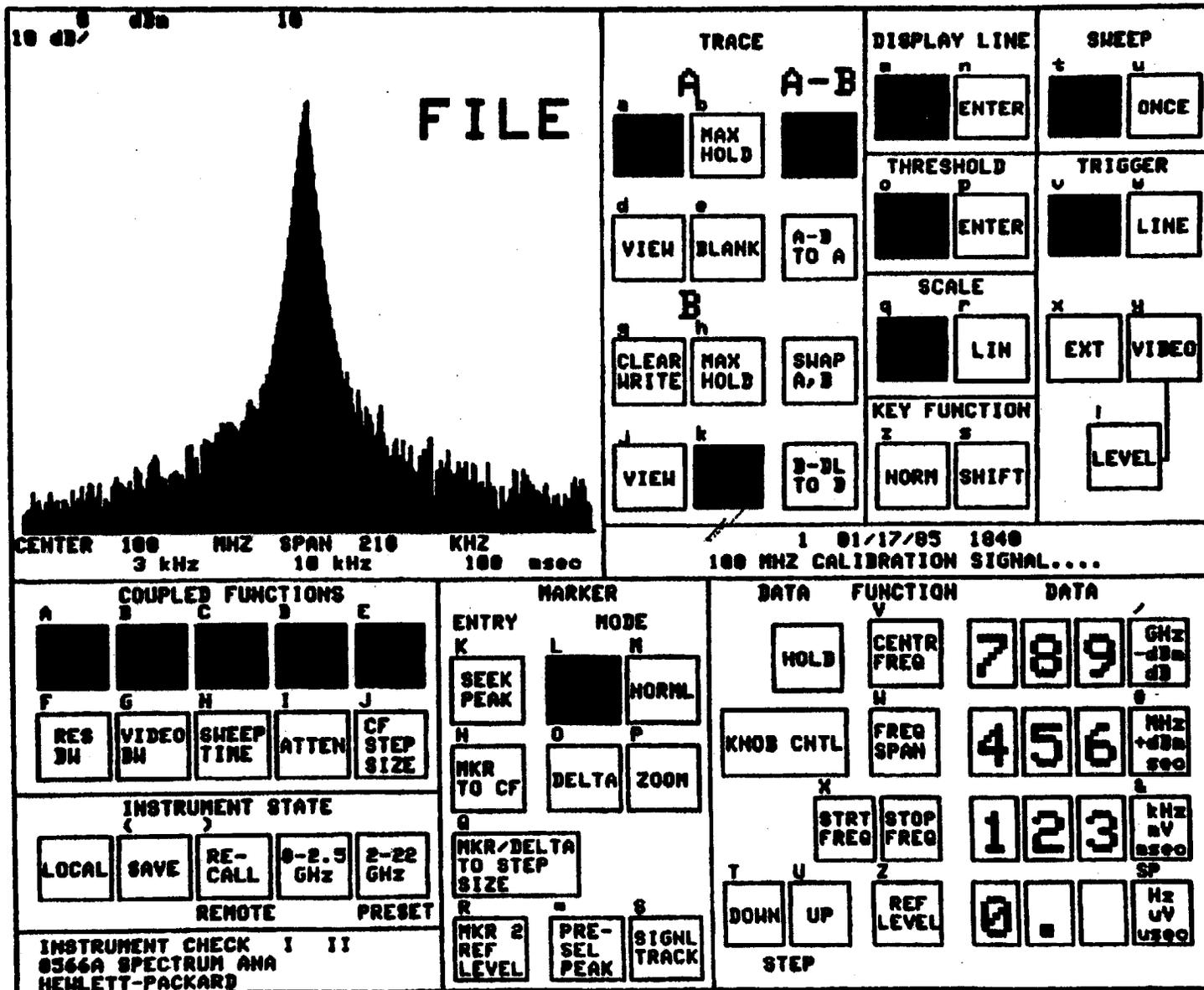


Fig. 14 Spectrum Analyzer Display

Table 1 GPIB Module - Interrupt Vector Assignment

Vector Number	Address (HEX)	Space	INTR Level	Assignment
64	100	SD	0	
65	104	SD	1	
66	108	SD	2	HINT
67	10C	SD	3	EOP1
68	110	SD	4	EOP2
69	114	SD	5	GPIB
70	118	SD	6	15 Hz
71	11C	SD	7	

Note: Interrupt levels are assigned in the interrupt controller chip (Am9519A) with Level 0 being the highest.