FT-68 M USER'S MANUAL

Forward

TECHNOLOGY INCORPORATED

Doc # 88-0002-00 REVA

<u>FT-68M USER'S MANUAL</u>

88-0002-00 Rev. A

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

SPECIFICATIONS

Physical

Width	:	12.00 in (30.48cm)
Height	:	6.75 in (17.15cm)
Depth	:	0.45 in (1.14cm)
Weight	:	18.0 ounces (512 gm)
Shipping Weight (without manual)	:	23.0 ounces (654 gm)
Form Factor	:	IEEE P-796

Environmental

Operating Temperature	:	0° to 50° C
Storage Temperature	:	-10° C to 70° C
Relative Humidity	:	90% without condensation

FT-68M Electrical Characteristics

5V$\pm$ 5%
3.1A nominal

System Clock

FT-68M	8.0 MHz $\pm$ 0.01%
Optional	9.8304 MHz $\pm$.01%

Connectors

Bus	86 Pin, 0.156 in center (0.4 cm)
Serial I/O	50 Pin Header

Electrical Interface

P-796	TTL Compatible
Interrupt Requests	TTL Compatible
Serial I/O	RS-423 Compatible

Processor

Motorola 68000 or equivalent
Direct addressing to 8 Mbyte of memory
Byte, word and long word data
16 individual 32-bit registers

Serial Communication Characteristics

Asynchronous	:	5 to 8 bits, 1, 1-1/2 or 2 stop bits
Synchronous	:	5 to 8 bits, internal or external sync. Automatic sync. insertion. CRC-16 generation/checking
Bit Synchronous	:	SDLC/HDLC Automatic sequence generation/detection Bit insertion/deletion. CRC generation/checking
Baud Rates	:	300, 1200, 2400, 4800, 9600.

1.0 Overview

The FORWARD Technology FT-68M is an IEEE P-796 Multibus® compatible high performance micro-computer mounted on a single printed circuit board. It provides system designers with a full performance implementation of the 68000 micro-processor which can be used as a stand-alone processor or for true parallel processing in multiprocessor environments.

The FORWARD Technology FT-68M combines a 68000 CPU, RAM, PROM, and Input/Output on a single board. Among its highlights are:

- an 8-MHz 68000 CPU, or an optional 10 MHz CPU;
- two-level, multi-process memory management;
- full-speed operation with local RAM (no wait states);
- up to 256K bytes of on board dynamic RAM with byte parity;
- up to 32K bytes PROM;
- dual USART, one 16-bit input port, five 16-bit timers;
- full Multibus Multimaster capabilities;
- single 5Volt power requirement.

LAY OUT OF THE FT-68M BOARD

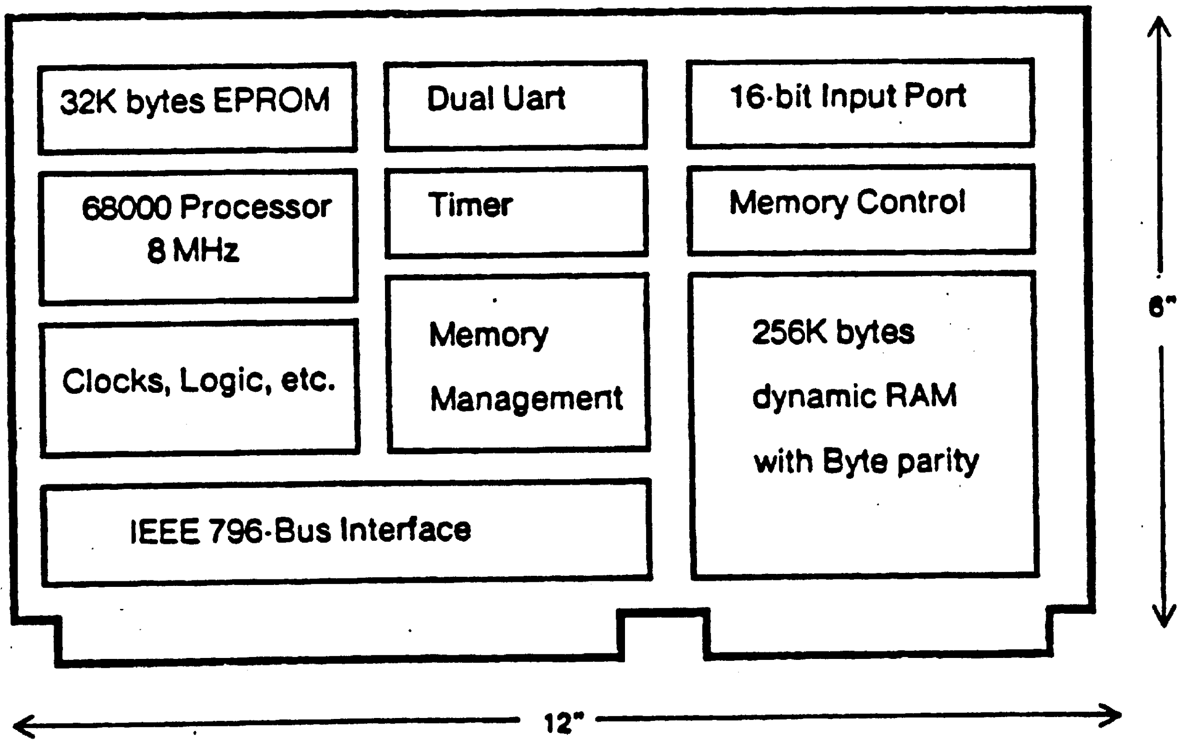


FIGURE 1-1

1.1 Central Processing Unit

The FT-68M is based on the 68000 processor, a high performance microprocessor with a 32-bit architecture and a large, uniform memory space. The 68000 features 16 32-bit registers. Eight registers are used for addresses and eight are used for data. Three major data sizes (byte, word, and long word) are used for data. The currently available implementation of the 68000 uses a demultiplexed 16-bit data bus and 24-bit address bus.

The FT-68M has been designed to achieve full performance from the 68000 processor. The board allows the 68000 to execute programs from onboard RAM at the fastest possible speed without wait states. This allows the 8 MHz 68000 to execute memory cycles approximately every 500 nsec. Accesses to devices on the Multibus are slightly slower, with cycle times being stretched by an amount equal to the device access time.

1.2 Bus Structure

The FT-68M has two busses: an internal synchronous bus for communicating with on-board memory and I/O options, and the Multibus system bus for referencing additional memory and I/O devices. On-board accesses do not require the Multibus, making the system bus available for use by other Multibus masters such as DMA devices. This also allows true parallel processing in a multiprocessor environment. The 68000 has complete access to the Multibus, but the Multibus has no access to the FT-68M's on-board memory or I/O.

Four sockets are provided for EPROM or ROM. The board accomodates 2716, 2732, or 2764 type PROMs, offering either 8, 16, or 32 KBytes of PROM.

1.3 On-Board Input/Output Capabilities

The FT-68M incorporates a dual USART, a multiple timer and a general purpose 16-bit input port.

The dual USART provides two high-speed serial I/O channels. One channel is configured to communicate with a terminal, the other can be connected to communicate with either a terminal or a host computer. Baud rates and communication modes are fully software programmable.

The timer provides five independent 16-bit counters/timers. Timers four and five are used as programmable baud rate generators for the USART. Timer three is dedicated to memory refresh. Timer two is optionally used for autoreset of the processor in case of an unexpected halt. This allows remote or stand-alone systems to automatically reboot themselves in case of catastrophic failure. Timer one is available for user applications and its output is unconnected.

The 16-bit parallel input port is intended to serve as an "option" port.

1.4 Multibus Capabilites

The FT-68M is IEEE P-796 (Multibus) compatible. The P-796 bus is an asynchronous bus, accomodating devices with various transfer rates while maintaining maximum throughput. A timeout is provided to abort Multibus cycles if the addressed device does not respond within 15 microseconds (12 microseconds for the 10 MHz option). Note: however, that the highest performance is achieved by operating from on-board memory. Accesses to the Multibus stretch the 68000 cycle by an amount equal to the device access time.

1.5 On-Board Capabilities

The FT-68M contains up to 256K bytes of dynamic RAM with byte parity and up to 32K bytes EPROM/ROM.

The RAM consists of 64K-bit dynamic RAM chips. The smallest configuration is 128K of RAM plus byte parity. The full population of the board provides 256K bytes of RAM with byte parity. RAM is refreshed in software by a non-maskable interrupt routine that executes every two milliseconds for a period of eighty microseconds (64 microseconds on the 10 MHz option).

Multibus devices must be able to support a service latency of 80 (or 64) microseconds max.

Using the 20 address lines of the standard Multibus, the FT-68M can address up to 1 megabyte of memory and input/output locations.

The FT-68M also has full multi-master capabilities that allow it to share the Multibus with several other processor boards or DMA devices. On-board arbitration logic automatically arbitrates access to the bus when an off-board cycle is executed, yielding to higher priority bus masters. Up to three masters can be in a system using priority arbitration and more can be accomodated using parallel priority arbitration.

The FT-68M can optionally provide Bus Clock, Constant Clock, and the Init Signals for the Multibus. Init is generated by an on-board precision voltage reference when the supply voltage falls below 4.75V.

1.6 IEEE-796 Bus Variations

The FT-68M varies from the 1980 IEEE-796 Bus Standards in the following ways:

P2-Connector:
The P2-Connector is reserved for memory expansion.

Interrupts:
Interrupts are handled by the 68000 in "auto-vector" mode.

INTA is not used. Interrupt requests are cleared by software.

Interrupt priorities are numbered corresponding to the 68000 definition. That is, INT1 is the lowest priority and INT7 the highest priority (non-maskable) interrupt.

INT0 is not supported (68000 only has 7 interrupt levels).

Byte Ordering:
The byte ordering on the Multibus corresponds to the Multibus definition, not the 68000 definition.

BCLK, CCLK:
Are driven to the Multibus with the 68000 system clock at 8 MHz BLCK and CCLK can be optionally disconnected. The 10 MHz option provides BCLK and CCLK at 9.8304 MHz.

INIT:
Is driven to the Multibus from the on-board precision voltage comparator. It can be optionally disconnected. INIT can be optionally driven from the Multibus overriding the on-board RESET.

CBRQ:
Not supported.

MRDC, IORC, METC, IOWC:
The drives on these lines is 20 mA instead of 32 mA.

XACK:

Timeout period is 15 microseconds (12 microseconds on the 10 MHz option) instead of 1 msec.

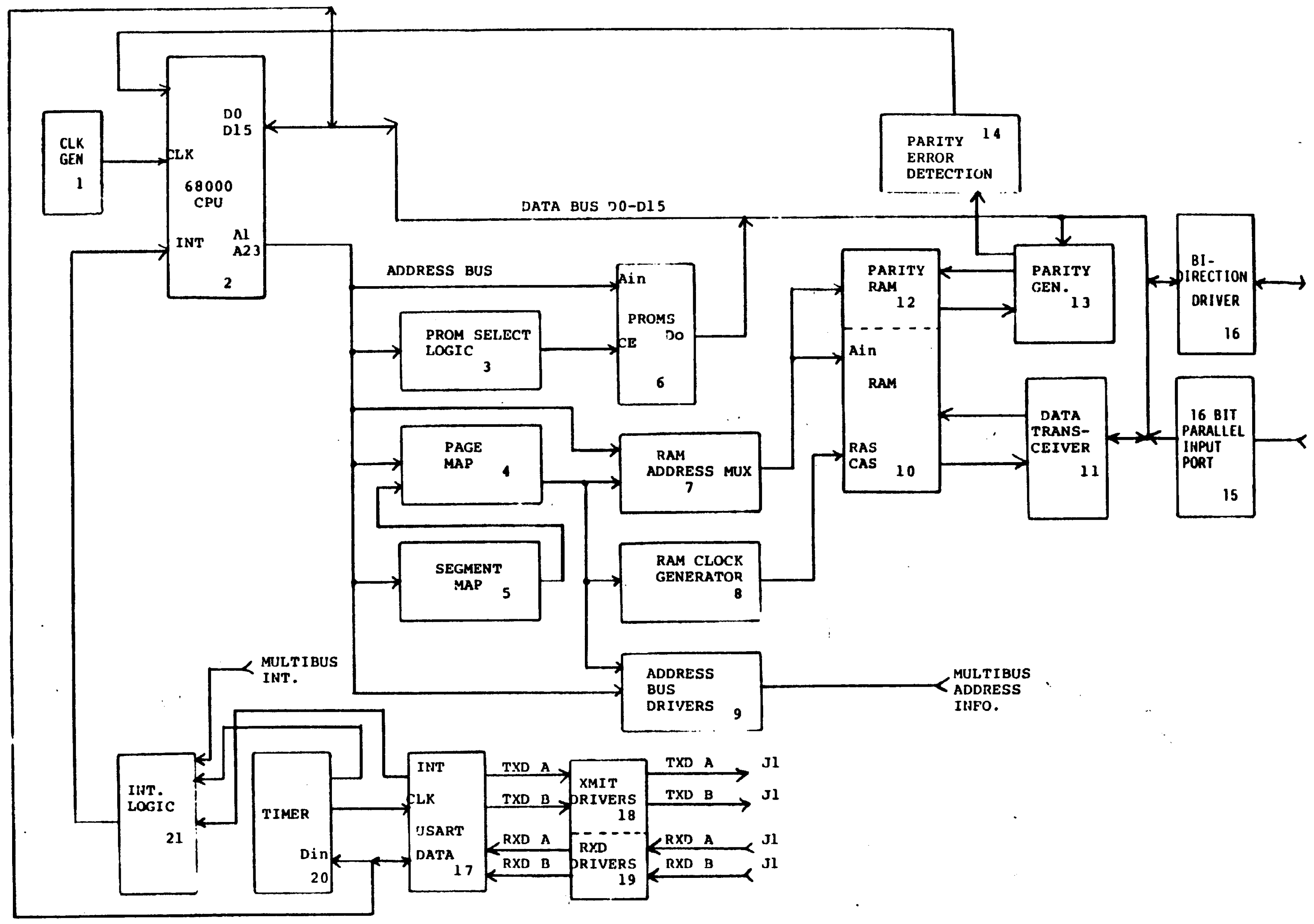
A short timeout guarantees that the 68000 processor can resume normal operation and satisfy real-time requirements without compromising system integrity. In a multi-master envirnoment, it also ensures that the bus is not locked out for excessive time. In other words, DMA requests to the bus are always serviced in a maximum of 15 microseconds.

2.0 System Architecture

FIGURE 2.1 is a block diagram of the FT-68M. Each block in the diagram is numbered. These numbers correspond with the descriptions below:

BLOCK #	NAME	FUNCTION
1	Clock Generator	This circuit provides all of the necessary clocks for the CPU, Timer, and associated circuitry.
2	CPU	This is the central processing/ arithmetic unit of the FT-68M. Based on the 68000 series Micro-processors, it has 16 bi-directional data lines and 23 address lines.
3	PROM Select Logic	This circuitry examines the address information from the CPU. If it falls within the range of PROM, it will generate the chip Enable Signal for the appropriate PROM.
4	Page MAP	The Page MAP forms part of the Memory Management Unit. The Page MAP converts address information from the segment MAP to a physical address which is used in conjuction with address information from the CPU to address RAM or devices on the Multibus.
5	Segment MAP	The segment MAP forms part of the Memory Management Unit. The Segment MAP translates address information from the CPU to a virtual address and sends this address information to Page MAP. The Segment MAP can be used to provide "Protection bits". These bits will determine what type of access will be allowed to a particular Memory location.
6	PROM	This block represents the PROM Memory on the FT-68M.

7	RAM Address Mux	This multiplexor presents 1/2 of the RAM address at RAS time and the other half at CAS time.
8	RAM Clock Generator	The RAM clocks (RAS & CAS) are generated by this circuitry.
9	Address Bus Drivers	These drivers provide address information to the Multibus.
10	RAM	This block represents the RAM Memory of the FT-68M. The standard board has 256K of RAM.
11	Data Transceiver	The data transceiver allows data to flow from/to the RAM Memory.
12	Parity RAM	This portion of RAM is used to store parity infomation generated by the parity generator.
13	Parity Generator	The parity generation circuit examines the data flowing to/ from RAM and generates a parity bit.
14	Parity Error Detection	The parity error detection circuit examines the output of the Parity generators. If an error is detected this circuit will notify the CPU.
15	Parallel Input Port	The parallel input port is 16 bits wide.
16	Bi-Directional Bus Drivers	These drivers handle all data communications between the FT-68M and the Multibus.
17	USART	The USART converts parallel Data from the CPU to Serial Data to be sent to an external device; it also converts serial data from an external device to parallel data for use by the CPU. The USART is programmable and is capable of using either asynchronous or synchronous protocols.


CLK GEN 1
68000 CPU 2
CLK
INT
D0 D15
A1 A23
DATA BUS D0-D15
ADDRESS BUS
PROM SELECT LOGIC 3
PAGE MAP 4
SEGMENT MAP 5
PROMS 6
Ain
CE
Do
RAM ADDRESS MUX 7
RAM CLOCK GENERATOR 8
ADDRESS BUS DRIVERS 9
MULTIBUS ADDRESS INFO.
PARITY RAM 12
RAM 10
Ain
RAS
CAS
PARITY GEN. 13
PARITY ERROR DETECTION 14
DATA TRANS-CEIVER 11
BI-DIRECTION DRIVER 16
16 BIT PARALLEL INPUT PORT 15
MULTIBUS INT.
INT. LOGIC 21
TIMER 20
Din
USART 17
INT
CLK
DATA
TXD A
TXD B
RXD A
RXD B
XMIT DRIVERS 18
RXD DRIVERS 19
J1
J1
J1
J1

18	Transmit Drivers	This circuit converts the TTL logic levels from the USART to EIA levels.
19	Receive Drivers	This circuit converts the EIA signal levels on the communications line to TTL levels for use by the USART.
20	Timer	The timer is a programmable device with 5 independent sections. Two of the sections are dedicated to providing the TCOM clocks to the USART. One section is used for Memory Refresh, another Section is utilized as a watchdog timer. The remaining section is for user applications.
21	Interrupt Logic	The interrupt logic examines interrupts from the TCOM, Timer, and Multibus circuitry and passes the interrupts on to the CPU.

2.2 Memory Management

The FT-68M provides up to 256 KB on board dynamic RAM with byte parity. Memory Refresh is handled by the microprocessor using a non-maskable interrupt routine that executes every two milliseconds for a period of eighty microseconds (8 MHz) or sixty-four microseconds (10 MHz).

Access to memory is controlled by the Memory Management Unit (MMU). The MMU uses a two-level memory management scheme that provides all the functions required to support powerful operating systems. These functions include relocation, protection, sharing, and dynamic allocation of main memory. All accesses by the CPU to memory, and to the Multibus I/O space, are translated and protected in an identical fashion.

(References to addresses within this test will be in hexadecimal notation where X stands for hexadecimal. Example X'1000'=decimal 4096.)

The page size is 2048 bytes, the segment size is 32 Kbytes, and up to 16 processes or contexts can be mapped concurrently. The MMU can simultaneously hold the translation for 1024 pages and 1024 segments divided among 16 processes. Thus the maximum logical address space for each process is 64 segments or 2 Mbytes, and the maximum physical address space that can be mapped simultaneously is 2 Mbytes as well.

Address translation is a two-step process: in the first step, the logical address from the CPU is translated into a virtual address via the segment map. In the second step the virtual address is translated into a physical address through the page map table. This entire translation, plus the checking of access rights, is performed without introducing wait states into the processor's operation.

Protection is associated with the segment map; each segment may have any one 16 different classes of access modes allowing read, write, and execute cycles in user and/or supervisor mode. The logical address space of each context is completely private. Sharing and interprocess communication are implemented in the virtual addresses, with segments from multiple contexts pointing to a shared virtual address.

Paging and the allocation of physical memory is done with the page map. A page map entry also indicates the physical address space in which a page is located, such as on-board or off-board memory. The page map further assists demand paging algorithms with reference and dirty bits. The page map is located at X'A00000' thru X'BFFFFF'.

2.2.1 Context Register:

In a Multitask environment it is important to be able to switch between processes quickly without having to reload all the translation state information of a particular process. The context register is a four-bit register that can switch between 16 separate portions of the segment map and thus between 16 tasks immediately. Only the four most significant bits (15, 14, 13, and 12) are used $\boxed{\text{NNNN|XXXX|XXXX|XXXX}}$ when loading the context register.

2.2.2 Segment Map:

The segment map is a 1024 entry table, indexed by the four-bit context register and the six most significant bits of the logical address space. The output of the segment map is six virtual address bits and four protection bits.

Each logical address space thus has up to 64 segments. Although segments are nominally 16 pages or 32 Kbytes large they can be as small as 1 page or 2048 bytes (by invalidating the undesired pages in the page map) or as large as 1024 pages or 2 Mbyte (by linking consecutive segment map entries).

2.2.3 Protection:

Each segment map entry carries 4 bits of protection information. The four bits are defined in the following table:

Protection Bits

	System	User	
0	---	---	No access
1	--x	---	
2	r--	---	
3	r-x	---	r - read
4	rw-	---	w - write
5	rwx	---	e - execute
6	r--	r--	
7	rw-	r--	
8	r--	rw-	
9	rw-	rw-	
A	rw-	r-x	
B	rw-	rwx	
C	r-x	r-x	
D	rwx	r-x	
E	rwx	--x	
F	rwx	rwx	
	System	User	

e.g.: .0FXX at location 0xC08000 is equal to segment map #1 allowed all access.

.08XX at location 0xC00000 is equal to segment map #0 with system read access only and user read and write access only.

Protection applies equally to the supervisor and the user state as well as all sixty-four segments.

2.2.4 Page Map:

In the page map, the virtual address from the segment map and the next four logical address bits from the CPU are translated into a physical address and a physical address space. Thus each segment accesses a block of 16 consecutive pages.

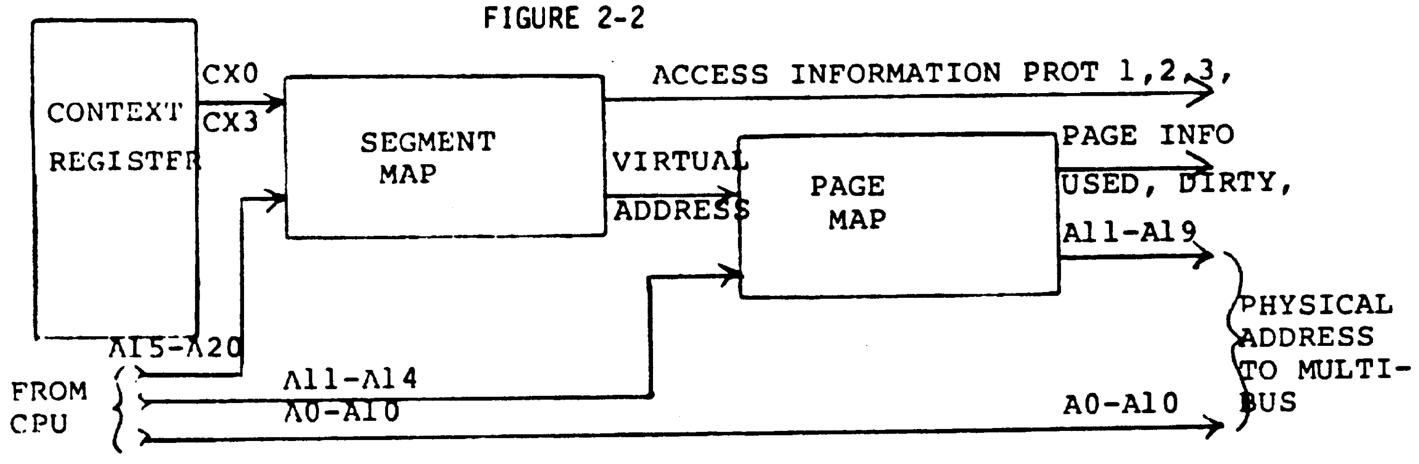
The output of the page map is 12 bits of physical address that is linked with the 11-bit byte address to form a 23-bit physical address. In addition, a page can be declared to be in on- board memory space, off-board memory space, off-board I/O space, or non-existent. A non-existing entry indicates an invalid page, causing an instruction abort. It is the responsibility of the memory management software to provide correct table entries for a particular system configuration.

2.2.5 Page Control:

In addition to the page mapping information, each page entry has two associated statistic bits, modified (dirty) and accessed, (used) that are set whenever that page has been accessed or has been written into. These bits are updated automatically on all cycles for which access has been granted by the protection mechanism.

FT-68M MEMORY MANAGEMENT BLOCK DIAGRAM

FIGURE 2-2



2.3 On-Board Input/Output:

Non-Bus accesses to the FT-68M are made via a dual universal synchronous/asynchronous receiver/transmitter. (USART) and a general purpose 16-bit parallel input port.

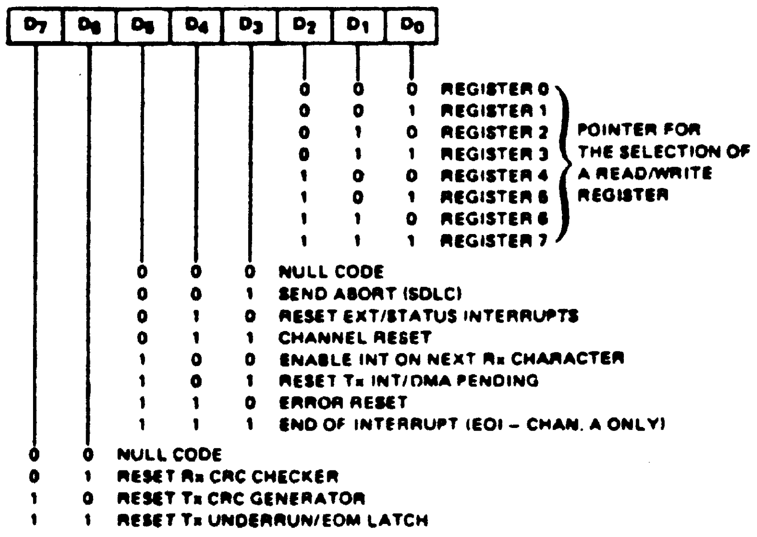
The dual USART provides two independent high-speed serial input/output channels. Received and transmitted data is available at connector J1. Refer to Figure 4-3 for pin assignments. The communications interface is factor set to operate at 9600 baud, asynchronously, and thus requires a terminal capable of using that data rate at initial system installation. Once communication with the FT-68M is established, the user may program other asynchronous, synchronous, or bit synchronous protocols at various data rates as required by his system. Refer to Section 4 in this manual for details on configuring the FT-68M for your particular communications needs.

Each channel of the USART is accessed via internal registers. These registers are known as the command register and the data register. Commands to, and status from each USART channel are handled via the command registers. Data to be transmitted, or data that has been received is handled via the data registers. The command register for channel 'A' is at memory location X'600002'. The data register for channel 'A' is at memory location X'600000'. The channel 'B' command register is at location X'600006' and the data register is at location X'600004'. The programmer programs the command register first (write register 0). Write register 0 will then point to the appropriate read or write register that the programmer desires to access. Once the programmer has accessed the desired register he is required to specify (via write register 0) which register he desires to access next.

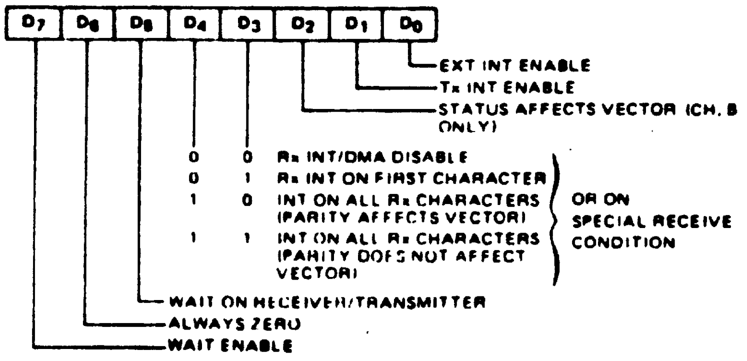
2.3.1 Command Register Format

WRITE REGISTER
BIT FUNCTIONS

WRITE REGISTER 0

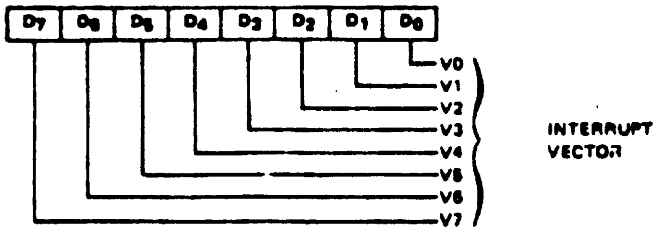


WRITE REGISTER 1

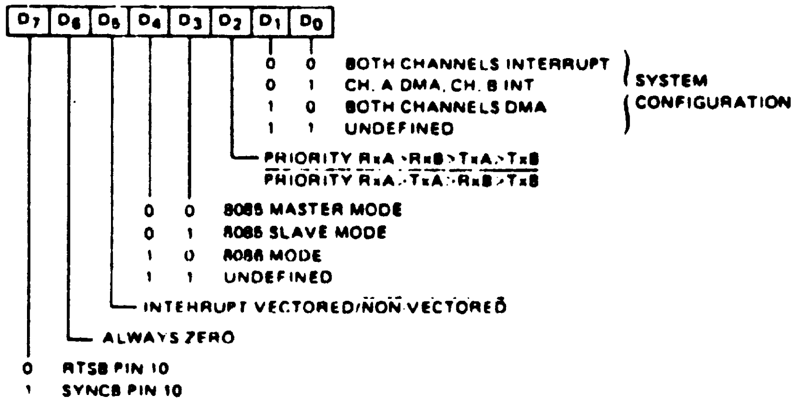


D7 D6 D5 D4 D3 D2 D1 D0
EXT INT ENABLE
Tx INT ENABLE
STATUS AFFECTS VECTOR (CH. B ONLY)
0 0 Rx INT/DMA DISABLE
0 1 Rx INT ON FIRST CHARACTER
1 0 INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR)
1 1 INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR)
OR ON SPECIAL RECEIVE CONDITION
WAIT ON RECEIVER/TRANSMITTER
ALWAYS ZERO
WAIT ENABLE

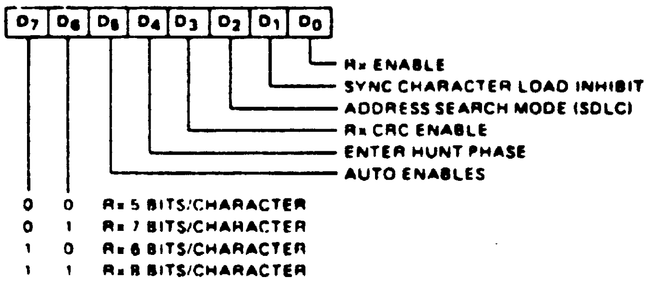
WRITE REGISTER 2 (CHANNEL B)



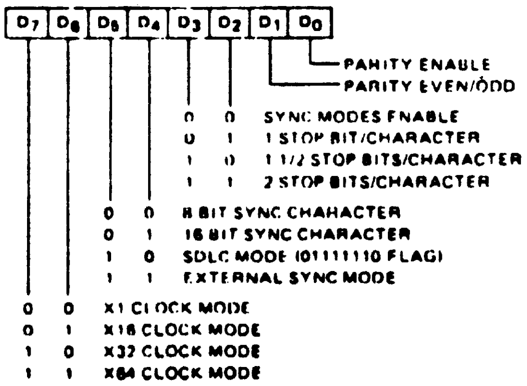
WRITE REGISTER 2 (CHANNEL A)



WRITE REGISTER 3

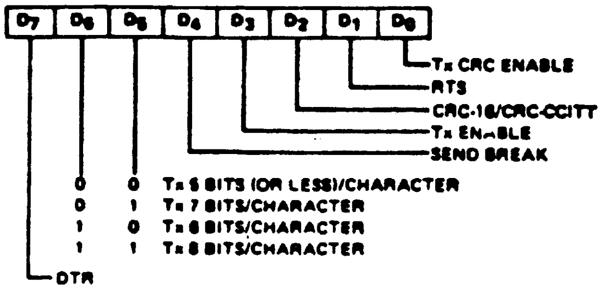


WRITE REGISTER 4

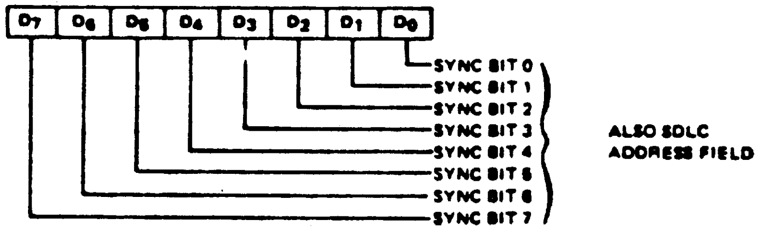


WRITE REGISTER BIT FUNCTIONS (CONT.)

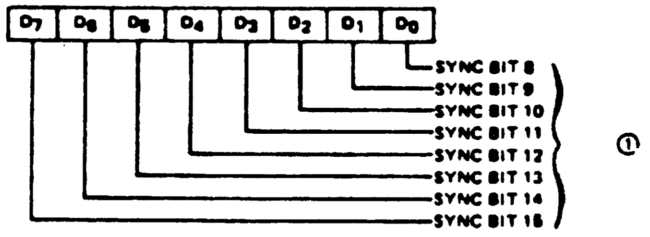
WRITE REGISTER 5



WRITE REGISTER 6



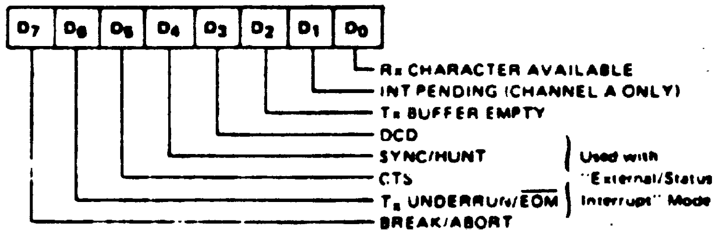
WRITE REGISTER 7



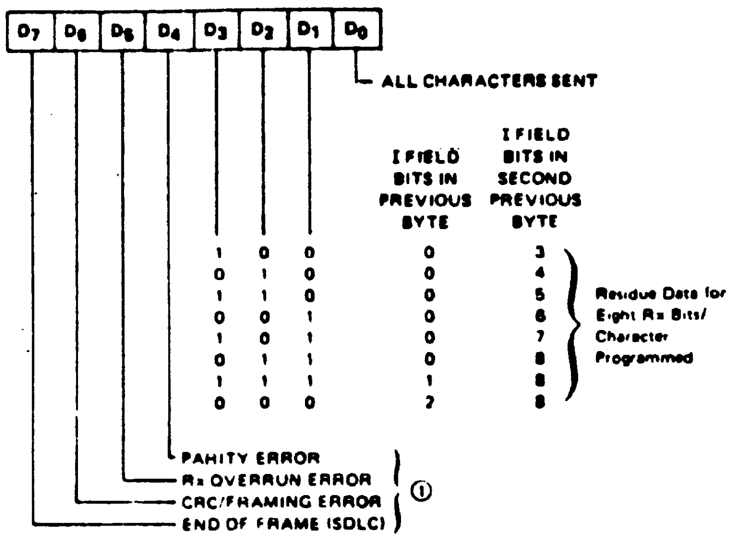
Note: ① For SDLC it must be programmed to "01111110" for flag recognition.

READ REGISTER BIT FUNCTIONS

READ REGISTER 0

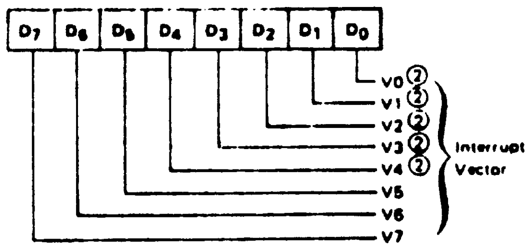

D7 D6 D5 D4 D3 D2 D1 D0
Rx CHARACTER AVAILABLE
INT PENDING (CHANNEL A ONLY)
Tx BUFFER EMPTY
DCD
SYNC/HUNT
CTS
Tx UNDERRUN/$\overline{EOM}$
BREAK/ABORT
Used with "External/Status Interrupt" Mode

READ REGISTER 1①


D7
D6
D5
D4
D3
D2
D1
D0
ALL CHARACTERS SENT
I FIELD BITS IN PREVIOUS BYTE
I FIELD BITS IN SECOND PREVIOUS BYTE
Residue Data for Eight Rx Bits/ Character Programmed
PARITY ERROR
Rx OVERRUN ERROR
CRC/FRAMING ERROR
END OF FRAME (SDLC)
①

D3	D2	D1	I FIELD BITS IN PREVIOUS BYTE	I FIELD BITS IN SECOND PREVIOUS BYTE
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

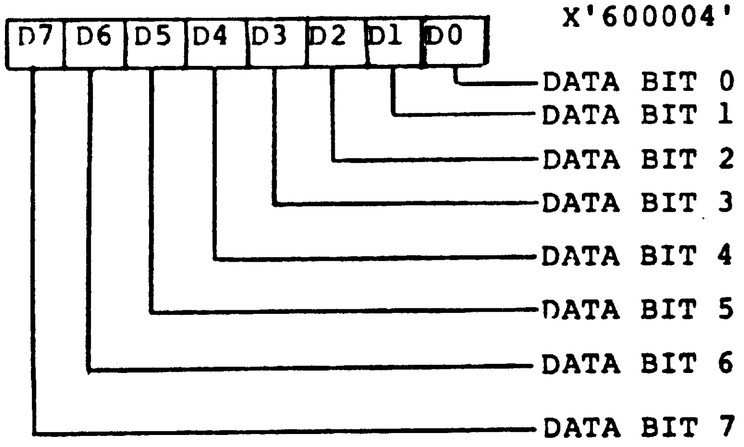
READ REGISTER 2



Notes ① Used with Special Receive Condition Mode.
② Variable if "Status Affects Vector" is programmed

DATA REGISTER

X'600000' CHANNEL A
X'600004' CHANNEL B



The general purpose 16 bit parallel input port is available by reading memory location X'E00000'. Information sent to the parallel input port by an external device is presented to connect J2. Refer to Figure 4-3 for pin assignments.

2.4 Timer:

The FT-68M utilizes a system timing control IC which provides five independant 16 bit counters/timers. Two of these timers (timer 4 & 5) are used in D mode to provide clocks to the USARTs for operation in the asynchronous mode. Timer three is dedicated to Memory Refresh. Timer 2 can be connected to the Interrupt 6 line and be used to automatically re-boot the system should a catastrophic failure occur. Timer one is available for user applications.

The system timing control I.C. is a programmable device which is accessed via memory locations X'800000' and X'800002'. Memory location X'800000' is the timer data register. Memory location X'800002' is the timer command register. Tables 1 and 2 show the count that must be loaded into the timer.

BAUD RATE	COUNT(Decimal)	COUNT (Hex)	% ERROR	FAST/SLOW
9600	13	000D	.16%	Fast
4800	26	001A	.16%	Fast
2400	52	0032	.16%	Fast
1200	104	0068	.16%	Fast
300	417	01A1	.08%	Slow
110	1136	0470	.03%	Fast

TABLE 2-1 8 MHZ COUNTER VALUES

BAUD RATE	COUNT(Decimal)	COUNT (Hex)	% ERROR	FAST/SLOW
19200	4	0004	0	N.A.
9600	8	0008	0	N.A.
4800	16	0010	0	N.A.
2400	32	0020	0	N.A.
1200	64	0040	0	N.A.
300	256	0100	0	N.A.
110	698	01BA	0.3%	Fast

TABLE 2-2 10 MHZ COUNTER VALUES

Should the user desire to run asynchronously at a baud rate other than those listed the count can be found using the following equation:

$$\text{Count} = \frac{\text{CLK (in Hz)}}{(\text{Baud Rate})(\text{CLK Multiplier})(2)}$$

CLK	= 4.0 MHz on the FT-68M 8 MHz pcb
CLK	= 2.46 MHz on the FT-68M 10 MHz pcb

*Note: The Clock Multiplier is what the user has programmed the USART FOR (X16, X32, X64, USART write register 4).

3.0 <u>OPERATION</u>

This section assumes that the user has correctly installed the FT-68M in a Multibus compatible chassis. An ASCII terminal capable of 9600 bit per second operating must be connected to serial port A. It also assumes that the user has installed a RESET switch. The latter item is not essential, but makes the procedure that follows much easier to perform.

3.1 <u>Initializing The FT-68M</u>

Insure that power is applied to the terminal connected to the FT-68M. Apply power to the Multibus chassis. This also applies power to the FT-68M. On power up, the FT-68M sends a message similar to the following to the terminal:

```
FTI GATEWAY MONITOR, VERSION 1-40000 BYTES OF MEMORY
```

If this message does not appear, check the terminal status and the terminal-to-FT-68M connection. If these check out correctly, toggle the RESET switch. The message should appear.

If all the connections and power are correct and repeated use of the RESET switch fails to generate the message, the FT-68M may be defective.

Successfully receipt of the message means that the FT-68M has performed a complete initialization procedure, and that the processor board can be assumed to be operating normally. The > symbol means that the PROM-resident monitor program is in COMMAND mode, and ready to communicate with the terminal. The Monitor has performed the following operations after a successful RESET:

- Normalizes all contiguous on-board RAM by writing data with correct parity into each address.

- Write the memory refresh routine and initial trap and vector settings into the first 2 Kb of RAM. Execution of the memory refresh routine is started.

- Sets the USART channels A & B to 9600 baud asynchronous ASCII protocol.

- Initializes and starts the counter/timer functions.

- Sets the Supervisor/Stack (SS) pointer to address X'001000'.

- Sets the User Stack (US) pointer to the top of available on-board memory.

- Sets the memory map Segment Table for context 0 and protection level 7 (Supervisor and User modes have read, write, and executive access to every segment).

- Sets the memory map Page Table. Existing pages of on-board RAM are mapped so that the physical and virtual addresses are identical.

- Trap vectors are set so that Monitor gains control of any exception interrupt.

Physical address space is divided into eight parts as follows:

X'00000' through X'1FFFFF': Mapped address space. The standard board has 256 Kb of on-board RAM, but may optionally have 128, 384, or 512 Kb. This space can also be mapped to the Multibus I/O or Multibus memory space.

X'200000' through X'3FFFFF': On-board PROM 0. Actual addresses are: X'200000' to X'201FFF' for 8 Kb
X'200000' to X'203FFF' for 10 Kb

X'400000' through X'5FFFFF': On-board PROM 1. Actual addresses are: X'400000' to X'401FFF' for 8 Kb
X'400000' to X'403FFF' for 16 Kb

X'600000' through X'7FFFFF': The on-board dual USART. The Channel A data register is at X'600000'; the Channel A command register is at X'600002'; the Channel B data register is at X'600004'; and the Channel B command register is at X'600006'.

X'800000' through X'9FFFFF': The on-board counter/timer. The data register is at X'9FFFFF' and the command register is X'800002'.

X'A00000' through X'BFFFFF': The page Map.

X'C00000' through X'DFFFFF': The Segment Map.

X'E00000' through X'FFFFFF': The Context Register.

3.2 Using The Monitor Program

The Monitor Program performs four basic functions: initializing the processor on power up or RESET, executing the memory refresh routine as described above, communicating with the user via console functions, and providing Emulator Trap service.

The console functions are routines that communicate with the user via the dual USART. All input/output is done using "busy-waits," and the code runs at the highest interrupt priority. Therefore, if a user program is interrupted with the ABORT switch or some other exception, the Monitor will run correctly unless the first 2 Kb of RAM has been damaged. If the user program is undamaged, it can be continued after the abort/interrupt and should be unaffected by the interruption except for the possible loss of some I/O data.

$\boxed{\text{W A R N I N G}}$

There are two rules for dealing with the monitor that ordinarily should not be violated.

1. User programs should not modify the context register directly. A special Emulator Trap instruction is provided to allow access to the Context Register, if required. This instruction is described later in the manual.

2. User programs should not write indescriminately into the first two kilobytes of RAM. It is legal for user programs to change exception vectors in Monitor dedicated RAM. However, no changes are allowed to the level 7 autovectors at address X'00007C' or to the User Interrupt Vectors located between addresses X'000100' and X'0003FF', inclusive. IF DATA IN THESE ADDRESSES IS CHANGED, THE MONITOR PROGRAM WILL FAIL. It will be necessary to RESET the FT-68M, and programs/data resident in RAM will be destroyed.

Console functions are invoked by commands issued in the format:

    Verb   Space* (argument) carriage return

The verb is always one alphabetic character, and may be either upper or lower case. Space* means that any number of spaces between the verb and argument/carriage return are ignored. Argument is normally either a hexadecimal number of a single upper or lower case letter. As indicated by the ( ), the argument may be optional. If an argument is present, it must be followed immediately by a carriage return (no space allowed) to start command execution. If a command is entered incorrectly, it can be changed or cancelled at any time before the carriage return key is pressed. Keying BACKSPACE or DELETE (or RUB) erases one character on the current command line. Keying Control U erases the entire command. The correct character/command may then be re-keyed.

NOTE: The symbol <CR> will be used throughout this manual to indicate depressing the carriage return key on the terminal used to communicate with the Monitor program.

In the description of command formats that follow the word open has special meaning. Anytime a memory location, map register, or CPU register is "opened", the name and contents of that address or register is displayed. If you key <CR>, the contents of the address/register remain the same and the Monitor program advances to the next address/register. If you key 0 <CR>, the contents of the address/register are set to zero and the Monitor program advances to the next address/register. If you key new hexadecimal value and <CR>, the new value is entered into the address/register and the Monitor program advances to the next address register. Keying Q <CR> or q <CR> returns the Monitor program to general command level, indicated by the > symbol.

For addresses, "next address/register" means the next byte or word of memory with a starting address larger than the current address. In other words, the command ascends through memory starting at the address given in the first command entered. For registers, "next address/register" means following the sequence: 68000 data registers D0 through D7; 68000 address registers A0 through A6; the Supervisor Stack (SS) pointer; the User Stack (US) pointer; the Status Register (SR); and the Program Counter (PC).

3.2.1 Monitor Commands

The Monitor commands are:

>A n<CR>	Opens 68000 address register n(where n is 0 to 6).
>B <CR>	Set a breakpoint. The Monitor prompts with the old breakpoint address. The user enters a new address, if desired.

>C [address]<CR>	Continue a program starting at the address given. If no address is specified, program execution begins at the current address.
>D n<CR>	Opens 68000 data register n(where n is 0 to 7).
>E [address]<CR>	Opens the work at memory location address. An odd address will be rounded down to the next even address.
>G [address]<CR>	Start executing a program at the address, if given. Otherwise, start executing at the current program counter location.
>H <CR>	Displays the Basic Commands Menu.
>I "mode" <CR>	Set the USART operation to "mode", where is A, B, or T. "A" means that Channel A to the terminal is operational. "B" means that Channel B to a host computer or terminal is operational. "T" puts the USART in direct channel (or transparent) mode between Channel A and Channel B. This allows the user at a terminal connected to Channel A to communicate with the host computer connected to Channel B directly. When communication with the Monitor is again required the user keys CRTL/SHIFT/6 or CRTL/↑ (up-arrow) followed by lower case C. The Monitor returns to A mode.
>K <CR>	This is a soft RESET. It resets the monitor stack and the default escape character. This command is useful after exceptions or other anomalous situations. Use of the K command may confuse the Monitor if a break point trap is set.

>L "Host Command"<CR>	This command is used to differentiate a host computer command from a Monitor command. The "host command" is sent to computer via USART Channel B. The Host must send a back slash character (\\) to the FT-68M in order to re-establish communications with the monitor after a file has been downloaded. Any data received by the FT-68M prior to receiving the backslash (\\) will be ignored.
>M m<CR>	Opens Segment Map register m.
>O [address] <CR>	Opens that byte at memory location address.
>P p<CR>	Opens Page Map register p.
>R <CR>	Opens the miscellaneous register in order, starting with the SS pointer, then the US pointer, SR, and finally the PC. The SS pointer may not be altered, and any attempt to do so is ignored by the monitor.
>S S-Record <CR>	This condition the Monitor to accept data from the host computer in S-record format. This is the only format that can be accepted from a host computer that is down-line loading programs. The Monitor will return one of three messages in response to the downline load: L for Length Error; K for Checksum Error; or Y for successful load.
>X Character <CR>	Sets the transparent mode escape character to "character". This allows the user to define a single character replacement for the control sequences given in the I command above.

Examples of the use of Monitor commands follow. The first sets memory address 1234 to 0F00. The second sets the contents of the 68000's data register 3 to 00CF. Underlined portions indicate the entries typed by the user. Non-underlined portions are generated by the Monitor.

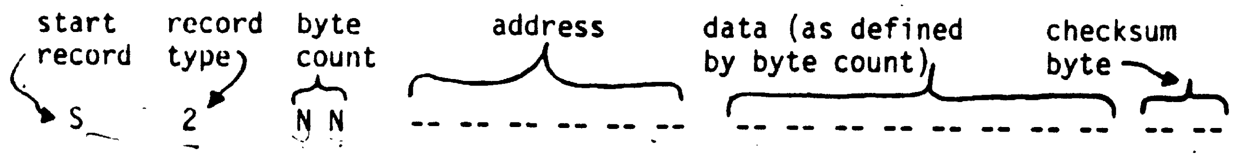
```
>e       1234<CR>
 001234:  23CF? 0F00 <CR>
 001236:  46FC? Q<CR>

>d   3 <CR>
 D3: 00000231? 00CF<CR>
 D4: 01303405? q <CR>
>
```

3.3 Loading Programs

The FT-68M is downline loaded using Motorola's "S" record format, so called because each type of record being with a byte containing the ASCII code for an S (Start of Record). An "S" record is a defined format used in transmitting and receiving programs and data. There are ten possible "S" record types, six of which are in active use, two which are defined but not in active use and two which are reserved. They are defined as follows:

S0	Header Record	Active
S1	16 bit address Data Record	Active
S2	24 bit address Data Record	Active
S3	32 bit address Data Record	Not Active
S4	Reserved	---
S5	Transmitted Data Record Count Record	Active
S6	Reserved	---
S7	32 bit address End of File/Execution address record	Not Active
S8	24 bit address End of File/Execution address record	Active
S9	16 bit address End of File/Execution address record	Active

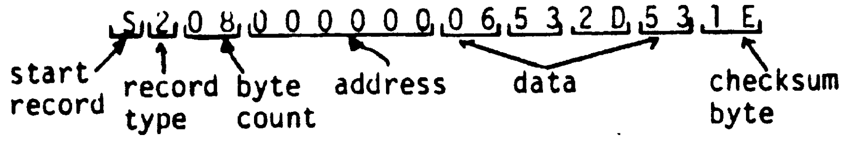
The "S" record format is shown below:



The "S" and the record type are represented directly in ASCII. The byte count, address, data, and checksum are represented in ASCII hexidecimal (i.e., two frames per data byte, with the most significant digit in the leading frame. The checksum is the 1's complement of the sum of all 8 bit data/address bytes from the byte count to the last data byte, inclusive and truncated to a byte.

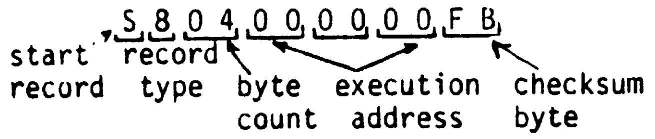
The FT-68M Monitor current supports only record types 2 and 8 (S2 & S8).

Example 1



```
checksum= 08
          00
          00
          00
          06
          53
          2D
          53
          --
          E1   1's compliment
               of E1 = 1E
```

Example 2



```
checksum = 04
           00
           00
           00
           --
           04   1's compliment of 04 = FB
```

The maximum "S" record length is 70 frames (a frame is defined as one-byte).

When using the S8 record type, an execution address of 000000 will cause no execution to occur; all other addresses are valid.

A typical file to be transmitted would be made up of the following sequence:

(LEADER) nx (S RECORD) (TRAILER)

The leader is a string of a minimum of 32 nulls followed by a carriage return, line feed, and null sequence. The trailer is a string of a minimum of 32 nulls.

A trailer must appear and goes at the end. Each S record is loaded into memory, starting with the address specified in the S 2 record, provided it passes the checksum test. The trailer serves two functions: to terminate reading the S-records and to load the program counter with the starting address. This is the mechanism for defining the entry point of a program. Program execution can then be started by issuing the G command.

3.4 C.P.U. Processing States

The 68000 is always in one of three processing states: normal, exception, or halted. The normal processing state is that associated with instruction execution; the memory references are to fetch instructions, operands, and to store results. A special case of the normal state is the stopped state which the processor enters when a STOP instruction is executed. In this state, no further memory references are made.

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient method for the processor to handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. A processor in the stopped state is not in the halted state, nor vice versa.

3.5 C.P.U. Privilege States

The processor operated in one of two states of privilege: the "user" state or the "supervisor" state. The privilege state determines which operations are legal, is used by the external memory management device to control and translate accesses, and is used to choose between the supervisor stack pointer and the user stack pointer in instruction references.

The privilege state is a mechanism for providing security in a computer system. Programs should access only their own code and data, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in the user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user state programs.

The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S-bit of the Status Register; if the S-bit is asserted (high), the processor is in the Supervisor State. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

3.6 Exception Processing

All exception processing is done in the supervisor state, regardless of the setting of the S-bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer. The processing of an exception occurs in four steps with variations for different exception causes. During the first step a temporary copy of the status register is made, and the register is set for exception processing. In the second step the exception vector is determined, and the third step is the saving of the current processor context. In the fourth step a new context is obtained, and the processor switches to instruction processing.

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts and the bus error and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from address errors or tracing. The trap (TRAP), trap on overflow (TRAPV), check register against bounds (CHK) and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, such as word fetches from odd addresses and privilege violations cause exceptions. Tracing behaves like a very high priority, internally generated interrupt after each instruction execution. Exceptions can be grouped according to their occurance and priority. The Group 0 exceptions are reset, bus error, and address error. These exceptins cause the instruction currently being executed to be aborted, and the exception processing to commence at the next minor cycle of the processor. The Group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but preempt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed.) The Group 2 exceptions occur as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while Group 2 exceptions have lowest priority. Within Group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within Group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T-bit asserted, the trace exception has priority, and is processed first. Before instruction execution resumes however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given below.

EXCEPTION GROUPING AND PRIORITY

GROUP	EXCEPTION	PROCESSING
0	Reset Bus Error Address Error	Exception processing begins at the next minor cycle.
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction.
2	TRAP, TRAPV, CHK, Zero Divide	Exception processing is started by normal instruction execution.

Bus Error Exceptions are generated by one of five conditions: (1) system space error; (2) segment map error; (3) page map error; (4) timeout error; and (5) parity error.

System Space Error: User processes are limited to two megabytes of address space. Larger addresses (A21 through A23 on the 68000 address bus) are used in the supervisor state to address on-board system facilities such as the memory management unit, the timer/counter, the dual USART, etc. Attempts to use an address greater than or equal to X'200000' in user mode will generate a bus error.

Segment Map Error: A segment map error occurs when the protection bits in the Segment Map do not allow the operation attempted. A Bus Error will be generated.

Page Map Error: A page map error occurs when an access to an invalid page is attempted. A Bus Error will be generated.

Timeout Error: Timeouts occur when off-board references to the Multibus are not acknowledged within approximately 15 microseconds. In most instances, a non-existent memory space or a non-existent device was addressed. A Bus Error will be generated.

Parity Error: On-board memory uses byte parity checking. When writing to local memory, odd parity is set for each byte. When reading from local memory, the parity of the 16-bit word (two bytes) is checked, and if it is odd (one of the bytes had even parity) a Bus Error is generated. Since the parity check is only completed at the end of a read cycle the 68000 can not abort the cycle in which the error occurred. Instead, the next cycle is aborted by a Bus Error Exception. The parity error condition is cleared by the next write cycle which is the start of stacking sequence in response to the Bus Error.

When a bus error occurs, the monitor prints the message:

```
>Bus Error:  address access - address at pc
```

The user can then read the access address in the program counter to determine when caused the exception. The user can determine the source of the Bus Error with the following alogrithm:

1. If the access-address was to system device (68000 bits A21 through A23 greater than zero), the only exception possible is that access was attempted in user mode, which is illegal.

2. If not 1, then check whether the operation violated the segment protection code.

3. If not 2, then check whether the page accessed is non-existent.

If none of the above caused the exception, the user must examine the bus/local bit in the page map.

4. If the access-address was to the bus, the access was aborted because of timeout (no bus acknowledgement within 15 microseconds).

5. If not 4, then a parity error occurred in the PREVIOUS memory cycle.

Address Error Exceptions occur when access to a word with an odd address is attempted. When an Address Error occurs, the Monitor prints the message.

```
>Address Error:  address access-address at pc
```

The user can then read the access-address in the program counter to determine what caused the exception.

3.6.1 Traps

Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from special instructions used to generate a trap. Traps are either Group 1 or Group 2 exceptions. When a trap occurs, the Monitor prints the message:

```
>Exception: XX

>
```

Where XX is a two letter code indicating the type of trap taken. Group 1 traps are characterized by the fact that exception processing begins before the next instruction. Group 1 traps recognized by the FT-68M, and the corresponding two letter codes are as below:

II	Illegal Instruction: an illegal instruction code was accessed.
Pr	Privilege Violation: an attempt was made to execute a privileged instruction while in user state.

L1-L6	Interrupt Autovector: an autovectored interrupt occured at one of levels 1 through 6.
U0	Unimplemented 0: this is a special case of illegal instruction. Instruction words with the pattern 1010 in bits 15 through 12 are assigned a separate exception vector. This provides processor access to user specified emulated interrupts. In this case, U0 is an FT-68M emulator trap.
U1	Unimplemented 1: same as above except the bit pattern is 1111. Also used for an emulator trap.
Un	Unassigned: trap was made to an unassigned vector.

Group 2 traps are characterized by the fact that exception processing is started by the execution of the trap instruction. Group 2 traps recognized by the FT-68M, and the corresponding two letter codes are:

ZD	Zero Divide: division by zero was attempted.
CH	Check: a CHK instruction faulted.
TR	Trap: a trap instruction was executed.
TV	Trap V: a trap on overflow (TRAP V) was taken.

In addition to the traps listed above, there are traps handled internally by the CPU that are not visible to the user. An excellent example ·s the memory refresh timer interrupt.

There are several exceptions handled especially by the Monitor. Two of these, Bus Error and Address Error, have already been described. The other exceptions occur in tracing programs and evoking emulator traps.

3.7 Tracing Programs

The Monitor provides primitive facilities for tracing program execution. The user must understand the program at the machine code level, and must have a symbol table listing of the program to be able to determine where each routine starts. The facilities used are Breakpoint Traps (BPT) and Trace Traps (TT).

Use of a BPT allows the user to run a program and return to the Monitor when execution reaches a certain point. Currently, the Monitor can only maintain one BPT at a time. To set a BPT issue the B command:

```
>B<CR>
```

The monitor prints the message

```
>BREAK XXXXXX?
```

Where XXXXXX is the location of the current BPT. To clear the BPT, type 0<CR>. Type<CR> only to leave the BPT set at the current location. Or, type a new address at which the new BPT is to occur (YYYYYY[CR]). This clears the old BPT and replaces it with the new one.

If the ABORT switch was used to access the Monitor before setting the BPT, continue program execution with the C command. Otherwise, start the program with the G command. In either event, execution should then proceed until the BPT is reached, at which point the Monitor prints:

```
>BREAK at XXXXXX.
```

The BPT is left alone, cleared, or altered as described above, and execution restarted using the C command. If an address is used with the C command, the BPT will not be reset, and execution continues from the address given.

If a program is loaded with the BPT set, the Monitor usually is able to detect thi'. If the K command (soft reset) is used, what happens when the BPT is taken is not predictable.

The Trace Trap provides the user with the ability to single step through a program, starting at a specific address. To set a TT, issue the R command and Use<CR> to step to the status register (SR). Inclusive - OR the contents of the SR with X'00008000', then exit the R command by type q<CR>. This sets the trace bit in the SR to trace mode.

3.8 Emulator Traps

The Monitor program provides services to user programs via Emulator Traps (EMT). An EMT is a convenient way of returning control to the Monitor which does not depend directly on the absolute addresses used. A program call to an EMT runs when the emulator sets an emulator trap code by setting bits 15 through 12 of the instruction word to 1010 or 1111. When the program encounters these codes, the result is essentially identical to a function call.

The services provided by EMT's fall into three categories: information EMT's; input/output EMT's; and memory management EMT's. Some of these, such as the memory management operations, are restricted to supervisor mode.

It should be noted here that the Monitor is written in the programming language C. The following paragraphs describe the C language calling sequence and descriptions for the EMT's. In general, if one of the functions described encounters an error condition, it returns the value 1. In particular, attempting to execute an EMT reserved for supervisor mode while operating in User mode will result in an error indication.

The FT-68M supports three information EMT's. The command formats and information provided are:

>int emt_ticks ( )<CR>	Returns the number of milliseconds since the Monitor was last booted. This is incremented each time memory is refreshed (at least every four milliseconds). The accuracy is not sufficient for time-of-day use.
>int emt_getmemsize ( )<CR>	Returns the size of the on-board RAM (plus optional additional RAM; if installed) in bytes.
>int emt_version ( ) <CR>	Returns the "version" of the ROM Monitor program; the most significant byte is the major version number; the next byte is the minor version number. For example X'00000105' corresponds to version 1.5.

>int emt_getcontext ( )<CR>	Returns the current value of the context register.
> emt_setcontext (cxt)<CR> >int cxt;<CR>	Sets the context register to cxt.

4.0 INSTALLATION

The FT-68M board physically conforms to the stands set down by the IEEE P-796 form factor. Consequently, the easiest method of installing a FT-68M board in a system is to construct the system using a Multibus standard chassis. Such chassis are commercially available from several sources. In addition, the IEEE P-796 standard provides sufficient information for the user to design a chassis to meet the specific application requirements.

If the user is integrating the FT-68M into an already existing chassis that does not conform to IEEE P-796 standards, the responsibility for insuring rigid mounting with proper clearance from other boards in the system is the user's. As a minimum, however, the board should be mounted on 3/4 inch centers parallel to other large boards. An adequate supply of cooling air, either convection or forced, is essential. The FT-68M consumes approximately 16 watts of power when running. Before performing the installation procedure described below, be sure to remove power from the system.

Inspect the board. If any components are mounted in sockets, insure that they are firmly seated in their sockets (EPROMs at U101 and U103, and possibly U105).

Insert the FT-68M P1 connector into the 86 pin Multibus connector (Viking 3 KH 43/9Am2 or equivalent). This connect should be wired for DMA device use -- that is, BPRN is forced low unless there is a multimaster environment.

If the memory expansion board (MEB), is being used, install it in the desired position in the chassis. Connect the FT-68M to the MEB with the cable supplied by FORWARD Technology. Connect the cable from P2 on the FT-68M to P2 on the MEB.

A female 50 pin leader connector (Augat at part number 110-50001-102 or equivalent) is required to/or access to the Serial I/O ports on connector J1. Port A and Port B are factory configured for EIA-RS423 level signals, using a 9600 baud asynchronous data rate.

To hook up a terminal via an RS-232C line, make connections as illustrated below. View the FT-68M board from the front, with the bus edge connector down. Viewed this way, J1 is the left-most 50-pin connector along the top of the board, and pin 1 is the left-most pin in the row numbered pins are the far row.

Don't forget that many terminals require jumpers for:

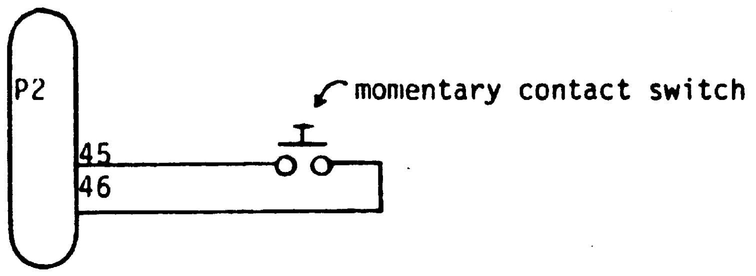
Request to Send - Clear to Send (4 to 5)

Date Terminal Ready - Carrier Detect (20 tp 8)

In order to operate correctly, the FT-68M receives and transmits at 9600 bits per second.

A switch to reset the FT-68M can be wired onto another 50-pin connector as shown below. Connect pin 45 through the switch to ground.

The pin numbering of J2 is the same as J1 described above, and all of the even pins are grounded on J2.



DEFAULT JUMPERING

The FT-68M comes with the 8218 bus arbitrator installed and jumpered to be the highest priority bus master. The bus signal BPRN is jumpered to ground on J901 (7 to 8). Also the FT-68M is strapped at J901 to drive the bus INIT line. If it is desired to use the bus INIT to reset the FT-68M, then cut J901 (3 to 4) and put a jumper on J901 (1 to 2). The FT-68M supplies on 8 MHz or 9.83 MHz to both BCLK and CCLK via J901 (5 to 6) and (9 to 10).

The dynamic memory refresh method of the FT-68M causes interrupt 7 to occur every two milliseconds and therefore interrupt 7 is disconnected from the bus by cutting J902 (1 and 2).

When the FT-68M is correctly hooked up and jumpered it will respond to power application with this message:

"FTI Gateway Monitor Version 1.2 -- 40000 bytes of memory"

The FT-68M supports RS-423 specifications for voltage driving at +5V. This will support and interface to existing RS-232C hardware. The rise and fall time (slew rate) of the FT-68M meets RS-232C specifications. To modify the slew rate for longer cables (e.g: 100'-4000'); the RS-423 slew rate of 10-90%must be maintained. To adjust the slew rate for Channel A (TxDA, RTSA and DTRA). Capacitors need to be added at the following locations (refer to Figure 4.1). Slew rate adjustment is available for Channel B only on Signal TxDB.

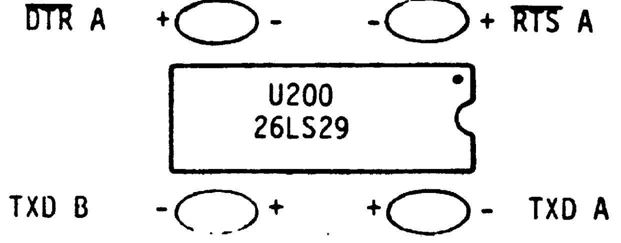


Figure 4.1

The capacitor value is dependent on the cable length and modulation rate. Refer to Figure 4.2 for information.

A capacitor of 50 pf will increase the rise time (slew rate) by 1 ns. The present rise time is 100 ns. A typical value for RS-423 specification is 3 μs, which will support either a cable length of 100 ft or modulation rate of 100K bps. If cable length is 50 ft or less, no modification should be done. If channel A cable length is to be increased and capacitors are to be added, the capacitors must be added to all of the lines used. (TxDa, RTSA or DTRA).

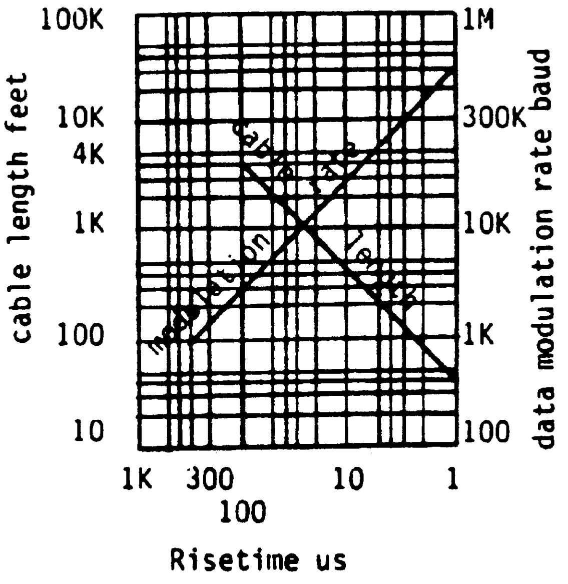
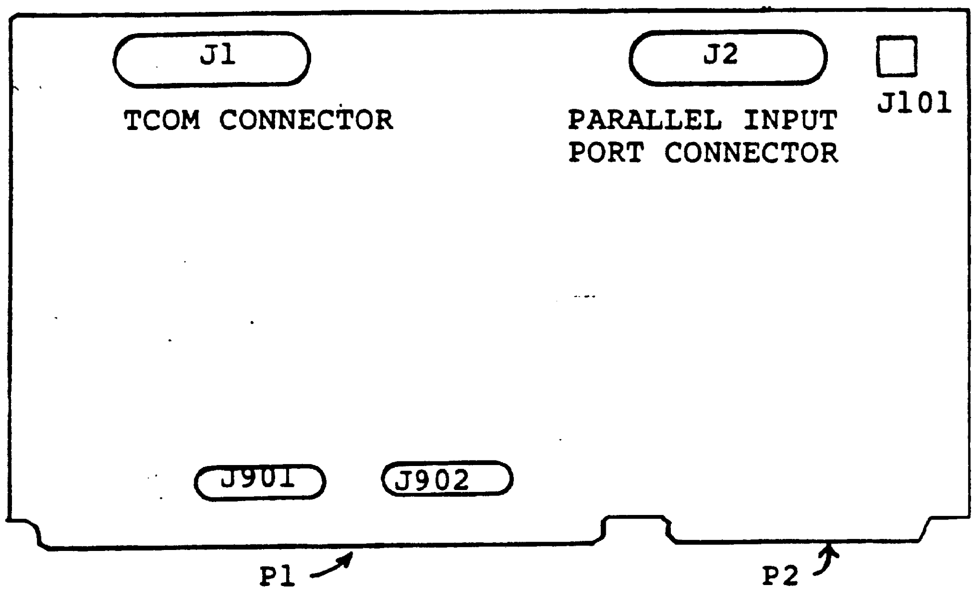


Figure 4.2



Signal Name		Pin #		
ransmit Data	(TXD)	03	Interface	J1
Receive Data	(RXD)	05	"A"	
Signal Ground	(GXD)	13		
ransmit Data	(TXD)	28	Interface	
eceive Data	(RXD)	30	"B"	
Signal Ground	(GND)	38		

Pin # to Pin #	Causes	J901 Bus Control
*3 - 4	Drives	INIT to Multibus
5 - 6	Drives	BCLK to Multibus
7 - 8	Grounds	BPRN for first master in chain
9 -10	Drives	CCLK to Multibus

Signal Name	J2	
J	Input Bit 0	-01
N1	Input Bit 1	-03
N2	Input Bit 2	-05
N3	Input Bit 3	-07
IN4	Input Bit 4	-09
N5	Input Bit 5	-11
N6	Input Bit 6	-13
IN7	Input Bit 7	-15
N8	Input Bit 8	-17
N9	Input Bit 9	-19
IN10	Input Bit 10	-21
IN11	Input Bit 11	-23
N12	Input Bit 12	-25
IN13	Input Bit 13	-27
IN14	Input Bit 14	-29
N15	Input Bit 15	-31
LR ABORT\	Idle contact of ABORT toggle	-39
SET ABORT\	Active contact of ABORT toggle	-41
LR INIT\	Idle contact of INIT toggle	-43
SET ABORT\	Active contact of INIT toggle	-45
REF\	Halt indicator output	-47
VCC	Halt indicator supply	-49
All even # pins are grounded		

Signal None	Pin# to Pin #		Interest Strapping J90
*B.INT7	1	2	(non-maskable interrupt)
B.INT6	3	4	(used by Timer 2)
B.INT5	5	6	(used by UART)
B.INT4	7-------	8	
B.INT3	9-------	10	
B.INT2	11------	12	
B.INT1	13------	14	
B.INT0	15------	16	(not used)

Signal Name	Pin #	Halt Line Connection J101
VCC	1	
M REF	2	

* NOTE: A line connecting adjacent Pin #s indicates factory strapping.