VMIVME-6016

16-CHANNEL INTELLIGENT ASYNCHRONOUS SERIAL CONTROLLER (IASC)

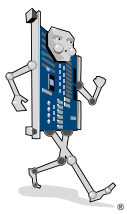
PRODUCT MANUAL

DOCUMENT NO. 500-006016-000 D

Revised May 7, 1997

VME MICROSYSTEMS INTERNATIONAL CORPORATION
12090 SOUTH MEMORIAL PARKWAY
HUNTSVILLE, AL 35803-3308
(205) 880-0444 FAX: (205) 882-0859
(800) 322-3616

COPYRIGHT AND TRADEMARKS



(I/O man figure)

UIOC®

WinUIOC®

(IOWorks man figure)

RECORD OF REVISIONS

REVISION LETTER	DATE	PAGES INVOLVED	CHANGE NUMBER
A	02/07/96	Release	96-0125
B	02/19/97	Cover, pages ii, 1-1, 1-2, 3-2, 3-3, and 5-5	97-0205
C	05/12/97	Cover, Pages ii and 5-4	97-0405

SAFETY SUMMARY

THE FOLLOWING GENERAL SAFETY PRECAUTIONS MUST BE OBSERVED DURING ALL PHASES OF THE OPERATION, SERVICE, AND REPAIR OF THIS PRODUCT. FAILURE TO COMPLY WITH THESE PRECAUTIONS OR WITH SPECIFIC WARNINGS ELSEWHERE IN THIS MANUAL VIOLATES SAFETY STANDARDS OF DESIGN, MANUFACTURE, AND INTENDED USE OF THIS PRODUCT. VME MICROSYSTEMS INTERNATIONAL CORPORATION ASSUMES NO LIABILITY FOR THE CUSTOMER'S FAILURE TO COMPLY WITH THESE REQUIREMENTS.

GROUND THE SYSTEM

To minimize shock hazard, the chassis and system cabinet must be connected to an electrical ground. A three-conductor AC power cable should be used. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the system in the presence of flammable gases or fumes. Operation of any electrical system in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS

Operating personnel must not remove product covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

DO NOT SERVICE OR ADJUST ALONE

Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT SUBSTITUTE PARTS OR MODIFY SYSTEM

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to VME Microsystems International Corporation for service and repair to ensure that safety features are maintained.

DANGEROUS PROCEDURE WARNINGS

Warnings, such as the example below, precede only potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

W A R N I N G

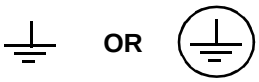
DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS SYSTEM. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.

SAFETY SYMBOLS

GENERAL DEFINITIONS OF SAFETY SYMBOLS USED IN THIS MANUAL

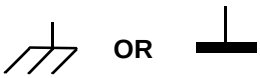
Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the system.

Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts are so marked).



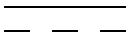
Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.

Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. Before operating the equipment, terminal marked with this symbol must be connected to ground in the manner described in the installation (operation) manual.

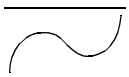


Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.

Alternating current (power line).



Direct current (power line).



Alternating or direct current (power line).

The WARNING sign denotes a hazard. It calls attention to a procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in injury or death to personnel.

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the system.

NOTE:

The NOTE sign denotes important information. It calls attention to a procedure, a practice, a condition or the like, which is essential to highlight.

VMIVME-6016
16-CHANNEL INTELLIGENT
ASYNCHRONOUS SERIAL CONTROLLER (IASC)

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

SECTION 4. PROGRAMMING (Concluded)

Page

TABLE OF CONTENTS (Continued)

SECTION 5. CONFIGURATION AND INSTALLATION (Concluded)

Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

TABLE OF CONTENTS (Concluded)

LIST OF TABLES (Concluded)

APPENDIX

SECTION 1

INTRODUCTION

1.1 INTRODUCTION

The VMIVME-6016 is a single-slot serial port controller with 16 or 32 channels, on-board ring buffers for each channel in both directions, and an on-board 25 MHz no-wait-state 68020 processor. The processor handles all character I/O and buffering, with Channel Control Blocks (CCBs) in on-board memory. User buffers, either linear or ring, reside either on-board or in VMEbus global memory. The VMEbus interface is controlled by a VIC068 or optional VIC64 VMEbus Interface Controller.

1.2 FEATURES

a. 8 or 16 channels available in one VMEbus slot

b. Line parameters independently controlled for each channel by control block in memory:

- baud rate
- internal ring buffer size
- user buffer size and address
- user buffer type (linear or ring)
- flow control: XON/XOFF, Any/XOFF, User characters, RTS/CTS, None
- interrupt on EOB or user-defined character
- interrupt on receive time-out
- interrupt on flow control
- interrupt on transmit complete
- BREAK send/receive and duration
- interrupt vector and level

c. RJ12 front panel connectors

d. Signal levels RS-232 compatible

e. Channel signals: TXD, RXD, RTS, CTS, DCD, GND

f. Short I/O-accessed control blocks

g. Standard/extended/DMA-accessed user buffers

h. Programmable VMEbus address modifiers

i. Size-programmable on-board ring buffers

j. 128 Kbytes, 256 Kbytes, 512 Kbytes, or 1024 Kbytes total user buffer space

k. 68020 processor, 25 MHz or 32 MHz, no-wait-states

l. Baud rates: 50 through 38,400 bps, each channel independent

m. Programmable interrupt vector and level

n. VMEbus compatible

o. MA32:MBLT32 as Master, optional MBLT64 with VIC64

p. SADO32:SD32 as Slave, optional MBLT with VIC64

q. Front panel status indicator

r. Programmable slave address for extended/standard buffer locations

s. Jumper-selectable slave address for short I/O

t. Bus release: ROR, RWD, FAIR, RCLR

u. Jumper-enabled system controller functions

1.3 REFERENCE MATERIAL LIST

Refer to "The VMEbus Specification" for a detailed explanation of the priority interrupt bus. "The VMEbus Specification" is available from the following source:

VITA
VFEA International Trade Association
10229 N. Scottsdale Road
Scottsdale, AZ 85253
(602) 951-8866

The following application and configuration guides are available from VMIC to assist the user in the selection, specification, and implementation of systems based on VMIC's products:

TITLE	DOCUMENT NO.
Digital Input Board Application Guide	825-000000-000
Change-of-State Application Guide	825-000000-002
Digital I/O (with Built-in-Test) Production Line Description	825-000000-003
Connector and I/O Cable Application Guide	825-000000-006

SECTION 2

PHYSICAL DESCRIPTION AND SPECIFICATIONS

REFER TO 800-006016-000 SPECIFICATION

SECTION 3

THEORY OF OPERATION

3.1 OPERATIONAL OVERVIEW

The VMIVME-6016 is a single-slot serial port controller with 16 or 32 channels, on-board ring buffers for each channel in both directions, and an on-board 25 MHz no-wait-state 68020 processor. The processor handles all character I/O and buffering, with Channel Control Blocks (CCBs) in on-board memory. User buffers, either linear or ring, reside either on-board or in VMEbus global memory. The VMEbus interface is controlled by a VIC068 or VIC64 VMEbus Interface Controller.

The VMIVME-6016 is an SADO32:SD32 VMEbus Slave for setup purposes. The VMEbus processor must set a number of items within the VMIVME-6016, such as interrupt vectors and masks. Most importantly, the host processor must set up a CCB for each of the channels. These blocks specify line parameters, flow control method, interrupts, on-board transmit and receive ring buffer size, and off-board user buffer size and method. The CCB allows each of the 16 channels to be independently programmable. When the host processor sets the GO bit for a channel, the VMIVME-6016 autonomously operates according to parameters in the corresponding CCB.

The front panel has an RJ12 6-pin telecom jack for each channel. Each channel supports signal ground and five signals:

- Receive Data
- Transmit Data
- Request to Send
- Clear to Send
- Data Carrier Detect

3.2 RS-232 CONNECTIONS

Each of the 16 channels is a 6-wire subset of the usual RS-232 signals. The RJ12 connectors are shown on schematic sheet 14. As viewed from the front panel, the RJ12 appears as in Figure 3.2-1. Each channel has the pinout shown in Table 3.2-1.

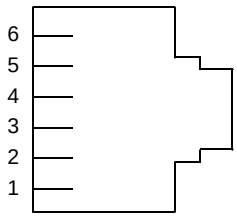


Figure 3.2-1. RJ12 Socket

Table 3.2-1. RJ12 Socket Pinout

Signal	RJ12	Meaning	Direction
CTS	1	Clear to Send	Input
GND	2	Signal Ground	
RXD	3	Receive Data	Input
TXD	4	Transmit Data	Output
DCD	5	Data Carrier Detect	Input
RTS	6	Request to Send	Output

M6016/T3.2-1

The VMIVME-6016 does not need ±12 V from the VMEbus backplane. The RS-232 levels are converted to and from TTL by the MAX244 chips shown on sheets 11 and 13 of the schematics. Using internal charge pumps, these chips generate their own ±12 V from the VMEbus +5 V. The charge pumps require the attached 1 µF capacitors. There are no jumpers or hardware configuration involved with the RS-232 interfaces.

3.3 CONTROL BLOCKS AND REGISTERS

The VMIVME-6016 emulates the register space with the top 256 bytes of on-board RAM accessed from the VMEbus by SADO16:SD32 (short I/O). Most of this space is taken up by 16 identically formatted control blocks, one for each of the 16 RS-232 channels. There is also a global register set for overall control and status.

3.4 VMEbus INTERFACE

The VMEbus interface is shown on schematic sheets 2, 5, 6, 7, and the top of 8. It uses a VIC as the core of the design. The CY7C964 slices save complexity of layout and logic design and support full master and slave VMEbus functionality including block-mode slave transfers. These transfers appear to the local bus as a chain of single cycles, so the CPU will not get locked out. See Section 3.5.1.

As a VMEbus slave, the VMIVME-6016 responds to A16 accesses to implement control and status registers with some of the on-board memory and to A24 and A32 accesses to implement on-board user buffers for each channel. The VMEbus slave specification is SADO32:SD32 (A32:A24:A16:ADO:D32:D16:D8(EO)). Supervisory only or both supervisory and nonprivileged accesses are allowed. Boards with the VIC64 option are additionally capable of MBLT64 VME64 accesses.

As a VMEbus master, the VMIVME-6016 can request the bus on any of the three BR levels, with AM code, BR level, and data width under control of programmable registers. The master specification is MA32:MBLT32 (A32:A24:A16:D32:D16:D8(EO):BLT)). The VIC64 option adds MBLT64.

As system controller (if installed in VMEbus slot 1), the VMIVME-6016 provides a full-function three-level arbiter, an IACK daisy-chain driver, a programmable BERR timer, a SYSCLK driver, and a SYSRESET* driver.

See Sections 3.5.3, 3.5.5, 3.6, and Sections 4 and 5 for details.

3.5 LOCAL BUS

A 32-bit data bus supports the ROM, RAM, 68020 CPU, UARTs, control and status registers for the VMEbus interface, and a buffer for reading the short I/O configuration jumpers. See Table 3.5-1 for the overall map for the local bus.

Table 3.5-1. Local Bus Map

Offset	Description
$000000 - 03FFFF	RAM
$040000 - 1FFFFF	Expansion RAM
$200000 - 27FFFF	VIC VMEbus Interface Controller
$280000 - 2FFFFF	LED control, jumpers, etc.
$300000 - 37FFFF	UARTs
$380000 - 3FFFFF	ROM

M6016/T3.5-1

The on-board CPU can make VMEbus master accesses by setting a direction control bit (TOLOCAL in the $280000 space) and then entering user mode, as opposed to supervisor mode. If the direction is TOLOCAL = 1, for example, a read accesses VMEbus space no matter what the address, and a write accesses local RAM. The TOLOCAL register is in the glue logic (see Section 3.5.6). The VMEbus cycles can be single cycle, bursts, or block transfers, depending on the programmable registers in the VIC chip.

3.5.1 CPU

The CPU (see schematic sheet 3) is a 25 or 32 MHz 68020. The CPU controls all other devices on the local bus and is responsible for transferring all UART data and status. It runs extensive tests on most of the hardware at power up. Additional diagnostic routines support production test. At the highest UART bit rate (38,400 bits/second), there are some throughput restrictions. The CPU has to process a data byte from a channel about every 15 microseconds if all are active at once in one direction. Bidirectional 38,400 bits/second on all channels simultaneously is not possible.

3.5.2 ROM

The VMIVME-6016 firmware resides in a 64 Kbyte ROM (27C512), on the lower half of schematic sheet 8. On power up or hard reset (by momentarily shorting J1), the firmware causes the CPU to run a test of the RAM. After this test, the whole content of the ROM is copied to the private section of RAM, the upper half. At least 64 Kbytes of private RAM are left for stacks and so forth. For the first eight byte-wide local bus cycles, the ROM is mapped to local bus zero (0). Afterwards, it is mapped at local $380000 and up allowing the CPU to get its initial program counter and stack address correctly. Once the CPU completes the power-on tests, it sets up defaults in the register space and begins waiting for the host to set up control blocks.

3.5.3 RAM

A 256 Kbyte static RAM, U16 on sheet 9, provides all of the space needed for the A16 space registers, local work area and code, and on-board user buffers. It is a 35 ns static RAM module, organized 64 Kbytes x 32 bits. See Table 3.5.3-1 for a map of the RAM.

Table 3.5.3-1. RAM Map

Offset	Description
$000000 - 01FFFF	On-board user buffers, if used
$020000 - 03FEFF	CPU private, and ring buffers
$03FF00 - 03FFFF	A16-space registers

M6016/T3.5.3-1

The RAM can be expanded to 512 Kbytes, 1 MB, or 2 MB, but it would only provide for larger on-board user and internal buffers. (See Section 4.2.15 for details.)

When the VMIVME-6016 is used in slave mode (on-board user buffers), the user buffers (linear or ring) appear in ascending order by channel, starting at the VMEbus address given in one of the global registers. This space is mapped into the lower half of the on-board RAM. The CPU's private memory (stacks, interrupt vectors, primary ring buffers, global variables, and code) begins halfway through the on-board RAM and ends just below the area taken by the A16 space registers.

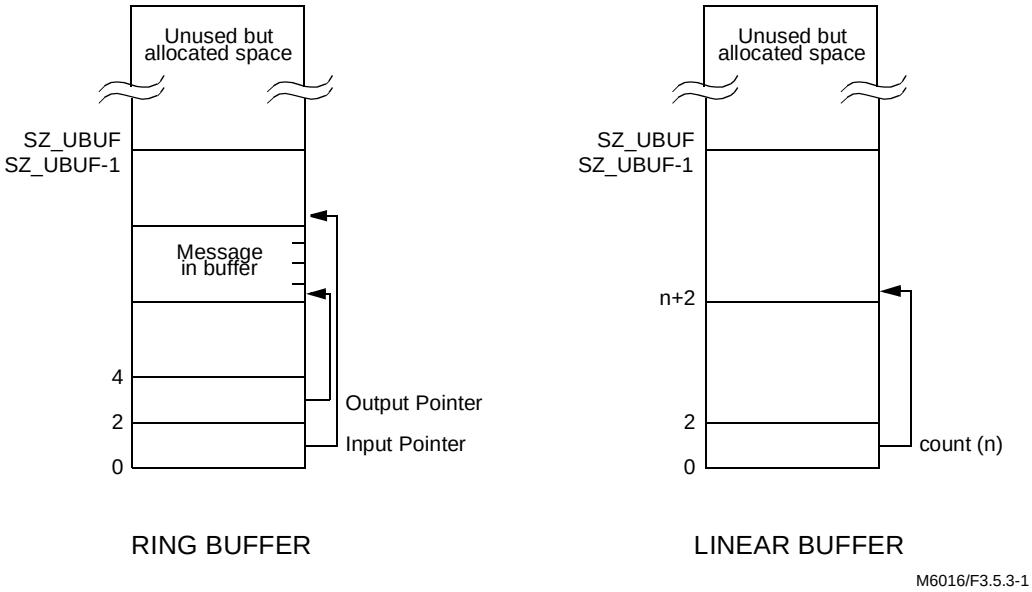


Figure 3.5.3-1. Ring and Linear Buffer Diagrams

3.5.4 UARTs

Two Phillips 2698B octal UART chips provide a total of 16 channels (shown on schematic sheets 10 and 12). Receive Ready signals from each channel can interrupt the on-board CPU with sixteen unique vectors. There are eight more vectors for pairs of channels for lower-priority events, such as Transmit Ready, Break, Overrun, and Parity Error. The TTL level serial signals get converted to and from RS-232 levels by five Maxim MAX244 self-powered transceiver chips.

3.5.5 Jumpers

The CPU can read the ten jumpers at JP3 on sheet 8 of the schematic by asserting the JMPRDN signal (see Section 3.5.6). There is no other connection to the jumpers except through the buffers to the Local Bus. The CPU uses the upper eight jumper locations to program the CY7C964 connected to VMEbus A[15..8] (sheet 7) to map VMEbus A16 slave space. The ninth jumper location is used by the CPU to

program the VIC chip slave decode control registers for either supervisor only or for both supervisor and nonprivileged. It is not possible to configure the VMIVME-6016 for nonprivileged only slave space. This jumper controls only the AM decode for A16 space; the A24 and A32 buffer space is programmable, but also for the same two choices (no nonprivileged only). The CPU uses the tenth jumper location to control the System Controller features of the VIC chip.

3.5.6 Glue Logic

The glue logic that provides the local bus memory mapping and the local bus read/write/select signals is in an 80-macrocell PLD-like FPGA at U9 on sheet 4. This is an Intel FX780, chosen because of its uniform timing and global routability.

3.6 VMEbus TIMER

The VMIVME-6016 has a hardware timer that counts 3.6864 MHz clock cycles while either DS0* or DS1* is asserted. Sheet 4, lower left, shows the timer. If about 50 microseconds elapse while the timer is counting, it asserts BERR* (if BTOEN is low). The BERR* negates about 300 ns after the DSx negate. Note that the DSx signals are isolated from the PLD by 220 ohm resistors which help reduce the DSx tracelengths and loading. See Section 4.2.5 for information on disabling the hardware timer.

The VIC chip supplies a programmable VMEbus time-out period (BERR* generator) ONLY if the board is configured for, and is in, VMEbus slot 1.

SECTION 4

PROGRAMMING

4.1 OVERVIEW

The VMIVME-6016 has 256 bytes of registers which appear in VMEbus A16 (short I/O) memory space. The Global Registers use the first 32 bytes of this space. The Channel Control Blocks immediately follow the Global Registers in short I/O space, channel 0 first (see Table 4.1-1). These registers should generally be accessed according to the size indicated, but the registers tolerate byte, word, longword, and unaligned access.

NOTE:

FOR A SHORT TIME AFTER POWER UP, THE VMIVME-6016 WILL NOT DECODE ACCESSES FROM THE VMEbus. THIS PERIOD IS ALWAYS LESS THAN TWENTY (20) MILLISECONDS. AFTER THIS TIME, ON-BOARD SELF-TESTS BEGIN TO RUN, DURING WHICH TIME THE VMIVME-6016 DECODES VMEbus A16 (SHORT I/O) ACCESSES AND SHOWS $FF IN THE POR SELF-TEST STATUS FLAG REGISTER. WHILE THE POR SELF-TEST STATUS FLAG REGISTER IS NONZERO, NONE OF THE REGISTERS BEYOND OFFSET $03 ARE VALID, AND VMEbus ACCESSES TO THE BOARD SHOULD BE KEPT TO A MINIMUM OR THE SELF-TEST WILL SLOW DOWN. THE SELF-TEST PERIOD IS LESS THAN TWENTY SECONDS (ABOUT FIVE SECONDS FOR THE NORMAL MEMORY CONFIGURATION, WHICH PROVIDES 128 KBYTES OF USER BUFFER MEMORY).

Table 4.1-1. VMIVME-6016 Register Map

Offset	Description
$00 - 1F	Global Registers
$20 - 2D	Channel Control Block 0
$2E - 3B	Channel Control Block 1
$3C - 49	Channel Control Block 2
$4A - 57	Channel Control Block 3
$58 - 65	Channel Control Block 4
$66 - 73	Channel Control Block 5
$74 - 81	Channel Control Block 6
$82 - 8F	Channel Control Block 7
$90 - 9D	Channel Control Block 8
$9E - AB	Channel Control Block 9
$AC - B9	Channel Control Block 10
$BA - C7	Channel Control Block 11
$C8 - D5	Channel Control Block 12
$D6 - E3	Channel Control Block 13
$E4 - F1	Channel Control Block 14
$F2 - FF	Channel Control Block 15

M6016

4.2 GLOBAL REGISTER MAP

The Global Registers control those features of the VMIVME-6016, and reflect those status conditions, that are independent of specific channels. This includes, for example, the slave mode base address and address space of the VMIVME-6016. Table 4.2.1 shows their offsets from the Board ID Register, the short I/O address of which is determined by a jumper field (see Section 5.3.1).

NOTE:

ANY BIT OR FIELD SHOWN AS ZERO (0) AND/OR RESERVED MUST BE SET TO ZERO (0).

Table 4.2-1. VMIVME-6016 Global Register Map

Offset	Name	Meaning	Read/Write
$00	BRD_ID	Board ID	Read Only
$01	STFLAG	Self-Test flag in ID Register	Read Only
$02	ROM_VER	ROM Version Code	Read Only
$04	CR0	Command Register 0	Read/Write
$05	CR1	Command Register 1	Read/Write
$06	GO	“GO” bits, ch[15..0] = GO[15..0]	Read/Write
$08	TX	“TX” bits, ch[15..0] = TX[15..0]	Read/Write
$0A	RX	“RX” bits, ch[15..0] = RX[15..0]	Read/Write
$0C	BREAK	“BREAK”, ch[15..0] = BREAK[15..0]	Read/Write
$0E	CR2	Command Register 2	Read/Write
$0F		Reserved	
$10	SZ_AM	Master’s Size and AM	Read/Write
$11	ST_PROC	Self-Test Procedure Register	Read/Write
$12	ER_MSK	Error Interrupt Control	Read/Write
$13	ER_VEC	Error Interrupt Vector	Read/Write
$14	BUFBASE	Base of Buffers (M) or Board (S)	Read/Write
$18	GST	Global Status Register	Read/Write
$1A	MAS_GRN	Master Granularity	Read/Write

M6016/T4.2-1

4.2.1 Board ID Register (BRD_ID)

The Board ID register is a read-only byte register. It always contains the value $3A, the VMIVME-6016 board identification code.

Table 4.2.1-1. Board ID Register Bit Map

Global Offset $00 BRD_ID (Read Only)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
0	0	1	1	1	0	1	0

M6016/T4.2.1-1

4.2.2 POR Self-Test Status Flag Register (STFLAG)

The Power-On-Reset (POR) Self-Test Status Flag register is a read-only byte register. Upon power up or VMEbus system reset, the VMIVME-6016 processor runs a self-diagnostic and reports the results to the GST register. A value of $FF indicates the board is still busy with its self-test, while a value of $00 indicates the test is complete. If the self-test fails, the Self-Test Fail bit in the GST is set. While the value is $FF, the board should be considered nonfunctional; the only register containing valid data besides this one is the Board ID register. Once the STFLAG register is clear, the board functions normally and all other registers are valid.

Table 4.2.2-1. POR Self-Test Status Flag Register Bit Map

Global Offset $01 STFLAG (Read Only)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
SELF-TEST STATUS							

M6016/T4.2.2-1

4.2.3 ROM Version Register (ROM_VER)

The ROM Version register is a read-only word register. The value in the lower byte of this register represents the revision number of the on-board firmware as two 4-bit BCD digits. The upper byte represents the hardware options installed.

Table 4.2.3-1. ROM Version Register Bit Map

Global Offset $02 ROM_VER (Read Only)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
INCONSIST_OPT	0	OLD_VIC_OPT	MBLT_OPT	SLOW_CPU_OPT	CHAN_OPT	MEM_OPT	

Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
FIRMWARE REVISION							

M6016/T4.2.3-1

Bit 15: Inconsistent Options (INCONSIST_OPT) – When set to one (1), this bit indicates that the ROM firmware is inconsistent with the hardware options.

Bit 14: Reserved – This bit is always zero (0).

Bit 13: Old VIC Option (OLD_VIC_OPT) – When set to one (1), this bit indicates that the VMIC silicon ID is too old.

Bit 12: MBLT Option (MBLT_OPT) – When set to one (1), this bit indicates that the VIC chip is VIC64 and MBLT is possible. A zero (0) indicates that the VIC chip is VIC68 and MBLT is **not** possible.

Bit 11: Slow CPU Option (SLOW_CPU_OPT) – When set to one (1), this bit indicates that the CPU is running at 25 MHz. A zero (0) indicates a 32 MHz CPU.

Bit 10: Channel Option (CHAN_OPT) – When set to one (1), this bit indicates that only 8 channels are available for use. A zero (0) indicates that 16 channels are available.

Bits 09-08: Memory Option (MEM_OPT) – These bits indicate the memory available on the VMIVME-6016 as follows:

Bit 09	Bit 08	User Buffer Area
0	0	128 Kbytes
0	1	256 Kbytes
1	0	512 Kbytes
1	1	1 Mbyte

4.2.4 Command Register 0 (CR0)

This register controls the major features of the VMIVME-6016. This register should be accessed according to Section 4.4.9.

Table 4.2.4-1. Command Register 0 Bit Map

Global Offset $04 CR0 (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
MASTER	SLV_OK	SLV_32	SUPONLY	TXBRK	SLFTST	RST_BRD	LEDN

M6016/T4.2.4-1

Bit 07: Master Mode (MASTER) – When set to one (1), this bit causes the VMIVME-6016 to run in master mode, in which the user buffers are off-board. In master mode, SZ_AM and BUFBASE point to a block of VMEbus memory that holds all the user buffers. When set to zero (0), the MASTER bit causes the VMIVME-6016 to run in slave mode, in which the user buffers are on-board. In this case, BUFBASE is the programmable VMEbus address of the on-board user buffers. In both cases, the user buffers are an array of possibly different-sized buffers. Each is allocated a fixed amount of memory whether or not it is all used: 1/64 of the total on-board RAM in the slave case and up to 32 Kbytes in the master case. They start with channel 0 input followed by channel 0 output, and finally ending with channel 15 output.

Bit 06: Slave Decode OK (SLV_OK) – When set to one (1), this bit allows VMEbus access to the user buffer memory on the VMIVME-6016. This bit should ordinarily be set to one (1) if the VMIVME-6016 is in slave mode (on-board user buffers), and be set to zero (0) otherwise. When this bit transitions from zero (0) to one (1), the VMIVME-6016 uses the contents of the BUFBASE to set the base address of the VMEbus space in which the user buffer memory will appear. After the slave address is set, BUFBASE can be used for other purposes.

Bit 05: Slave is A32 (SLV_32) – When set to one (1), this bit allows extended (A32) VMEbus access to the user buffer memory on the VMIVME-6016. When this bit is set to zero (0), the VMIVME-6016 responds to standard (A24) accesses, and the most significant eight bits of BUFBASE are ignored.

Bit 04: Slave decodes only Supervisor Address Modifiers (SUPONLY) – When set to one (1), this bit allows only supervisor accesses to the user buffer memory on the VMIVME-6016. When set to zero (0), it allows both supervisory and nonprivileged accesses. This bit has no effect on short I/O (A16) accesses to the VMIVME-6016. (There is a jumper for that function; see Section 5.3.1.)

Bit 03: Transmit BREAK Signals (TXBRK) – When this bit is set from zero (0) to one (1), a BREAK signal is sent on each active channel for which a bit is set in the BREAK register. Bits are automatically cleared in BREAK as the signals are sent, and when all the requested BREAKs have been sent, the TXBRK bit is automatically reset to zero (0).

Bit 02: Run Self-Test (SLFTST) – When this bit is set from zero (0) to one (1), the self-tests requested in the ST_PROC register are run in order of most significant ST_PROC bit to least significant ST_PROC bit. When the tests have run, or a failure occurs, the testing stops and the SLFTST bit is automatically reset to zero (0). The STBUSY bit of the GST should be one (1) until the self-test is complete. The ANYERR and STFAIL bits of the GST should be zero (0) if the self-test passed.

Bit 01: Reset Board (RST_BRD) – When this bit is set to a one (1), the VMIVME-6016 goes through nearly all the POR test sequence (excepting only the RAM shake) and presents itself as if power had just come on. Note that the board briefly "disappears" from short I/O space as it executes the POR tests. See Sections 4.1 and 4.2.2.

Bit 00: LED Control (LEDN) – When this bit is set to one (1), the LED on the front panel is off. Clear this bit to turn the LED on. The LED is off after the power-up (or reset) self-tests complete successfully. (The LED flashes on and off during these self-tests.)

4.2.5 Command Register 1 (CR1)

This register controls the BERR* timer and the arbitration mode. Bits not mentioned below must remain set to zero (0).

Table 4.2.5-1. Command Register 1 Bit Map

Global Offset $05 CR1 (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
0	0	0	0	ARBMOD	BERR_TO		

M6016/T4.2.5-1

Bit 03: Arbiter Mode (ARBMOD) – This bit has no effect unless the VMIVME-6016 is in the VMEbus slot 1 as the system controller and the SCON jumper is installed (see Section 5.3.2 for details). If the VMIVME-6016 is the system controller, a zero in this bit causes the VMIVME-6016 VMEbus arbiter to use a round-robin arbitration scheme, while a one (1) causes it to use a prioritized arbitration scheme.

Bits 02-00: SCON Mode VMEbus Timeout (BERR_TO) – When the VMIVME-6016 is system controller, this field sets the maximum allowed time before a BUS ERROR signal is issued in the absence of a DTACK for any VMEbus access. The maximum time codes mean:

Bit 2	Bit 1	Bit 0	Maximum Time
0	0	0	4 microseconds
0	0	1	16 microseconds
0	1	0	32 microseconds
0	1	1	64 microseconds
1	0	0	128 microseconds
1	0	1	256 microseconds
1	1	0	512 microseconds
1	1	1	Infinite

When the VMIVME-6016 is **not** the system controller, any value in BERR_TO other than binary 111 selects a nominal 50 microsecond timeout, and 111 disables the timer. In this case, the timer applies only to VMIVME-6016 accesses.

4.2.6 GO Bits Register (GO)

The GO register has a GO bit for each of the 16 channels. After the Channel Control Blocks and the Global Registers have been set up, setting the GO bit for a given channel to one (1) lets the VMIVME-6016 activate that channel. Activity for a channel stops if the corresponding GO bit is set to zero (0). Bit 15 (the MSB) corresponds to channel 15, bit 14 is for channel 14, and similar mapping goes through bit 0 (the LSB), which is for channel 0. Setting GO for a channel automatically clears TX and RX for that channel. This register should be modified using an indivisible RMW instruction. See Section 4.4.9 for details.

Table 4.2.6-1. GO Bits Register Bit Map

Global Offset $06 GO (Read/Write)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
Channel 15	Channel 14	Channel 13	Channel 12	Channel 11	Channel 10	Channel 09	Channel 08
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
Channel 07	Channel 06	Channel 05	Channel 04	Channel 03	Channel 02	Channel 01	Channel 00

M6016/T4.2.6-1

4.2.7 Transmit Request Bits Register (TX)

The TX register has a transmit request bit for each of the 16 channels. Provided the GO bit for a channel is set, setting the corresponding TX bit will request the VMIVME-6016 to accept data from the transmit user buffer and transfer it through the internal ring buffer to the transmitter. Bit 15 (the MSB) corresponds to channel 15, bit 14 is for channel 14, and similar mapping goes through bit 0 (the LSB), which is for channel 0. These bits can be understood as notifying the VMIVME-6016 that the user buffer contains new data to be transmitted. This register should be modified using an indivisible RMW operation. See Section 4.4.9 for details.

Table 4.2.7-1. Transmit Request Bits Register Bit Map

Global Offset $08 TX (Read/Write)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
Channel 15	Channel 14	Channel 13	Channel 12	Channel 11	Channel 10	Channel 09	Channel 08
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
Channel 07	Channel 06	Channel 05	Channel 04	Channel 03	Channel 02	Channel 01	Channel 00

M6016/T4.2.7-1

4.2.8 Receive Accept Bits Register (RX)

The RX register has a receive-accept bit for each of the 16 channels. Provided the GO bit for a channel is set, setting the corresponding RX bit notifies the VMIVME-6016 that the data in the user receive buffer has been accepted and that new data can be put into the user receive buffer from the internal ring buffer. Bit 15 (the MSB) corresponds to channel 15, bit 14 is for channel 14, and similar mapping goes through bit 0 (the LSB), which is for channel 0. Note that because there is no internal ring buffer in the on-board ring buffer case (UB_RING = 1, see Section 4.3.11), the RX bits for channels configured this way have no function and are ignored. This register should be modified using an indivisible RMW operation. See Section 4.4.9 for details.

Table 4.2.8-1. Receive Accept Bits Register Bit Map

Global Offset $0A RX (Read/Write)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
Channel 15	Channel 14	Channel 13	Channel 12	Channel 11	Channel 10	Channel 09	Channel 08
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
Channel 07	Channel 06	Channel 05	Channel 04	Channel 03	Channel 02	Channel 01	Channel 00

M6016/T4.2.8-1

4.2.9 Send Break Bits Register (BREAK)

The BREAK register has a bit for each of the 16 channels. After the Channel Control Blocks and the Global Registers have been set up, setting the BREAK bit for a given channel to one (1), and then setting the TXBRK bit, makes the VMIVME-6016 send a BREAK signal on that channel. The bit automatically resets to zero (0) when the BREAK signal starts. The BREAK is an RS-232 spacing level lasting for a number of milliseconds set by the BRK_DUR register in the Channel Control Block.

Table 4.2.9-1. Send Break Bits Register Bit Map

Global Offset $0C BREAK (Read/Write)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
Channel 15	Channel 14	Channel 13	Channel 12	Channel 11	Channel 10	Channel 09	Channel 08
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
Channel 07	Channel 06	Channel 05	Channel 04	Channel 03	Channel 02	Channel 01	Channel 00

M6016/T4.2.9-1

4.2.10 Control Register 2 (CR2)

This register controls the VMEbus request level, release mode, and fairness timeout for master accesses to the VMEbus.

Table 4.2.10-1. Control Register 2 Bit Map

Global Offset $0E CR2 (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
BRLEVEL		RELMOD		FAIR_TO			

M6016/T4.2.10-1

Bits 07-06: Bus Request Level (BRLEVEL) – This field controls the VMEbus bus request level for the VMEbus DMA accesses during transfers. The default value is binary 11, indicating a bus request level of 3, the highest value available.

Bits 05-04: VMEbus Release Mode (RELMOD) – This field determines the VMEbus Release Mode when the VMIVME-6016 is the bus Master. The Release Mode codes are:

Bit 05	Bit 04	Code	Description
0	0	RELROR	Release on Request
0	1	RELRWD	Release when Done
1	0	RELROC	Release on BCLR*
1	1	RELBCAP	Bus capture and hold

The default value for this field is binary 00, indicating Release on Request mode.

Bits 03-00: VMEbus Fairness Timeout (FAIR_TO) – The value in this bit field determines how many 2 microsecond intervals, if any, the VMIVME-6016 will wait before asserting its VMEbus request if another VMEbus master is already using the bus. This fair-request scheme of bus arbitration avoids the bus starvation that can occur when multiple bus masters exist, especially multiple masters at the same priority level. A value of zero (0) here disables the fairness timeout and a value of binary 1111 causes the VMIVME-6016 to wait indefinitely for the other bus master to relinquish control. Note that the actual timeout value may be up to 2 microseconds longer than programmed. The default value for this field is binary 0010, indicating a 4 microsecond timeout.

4.2.11 Master Size and Address Modifier Register (SZ_AM)

The Master Size and Address Modifier Register is a read/write byte. This register contains the information needed to read or write data directly to or from VMEbus memory.

Table 4.2.11-1. Master Size and Address Modifier Register Bit Map

Global Offset $10 SZ_AM (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
VMESIZE		ADDRMOD					

M6016/T4.2.11-1

Bits 07-06: VMESIZE – contains a two-bit code representing the data width of the VMEbus access. The codes are:

Bit 07	Bit 06	Code	Description
0	0	VME_SIZE_32	32-bit size as VMEbus master
1	0	VME_SIZE_16	16-bit size as VMEbus master
0	1	VME_SIZE_8	8-bit size as VMEbus master

Bits 05-00: ADDRMOD – directly corresponds to the VMEbus address modifier (AM) bits AM5-AM0, bit 5 corresponding to AM5 through bit 0 corresponding to AM0. Typical (hexadecimal) VMEbus address modifier values are:

Value	Description
$09	Extended (A32) Nonprivileged Data
$0A	Extended (A32) Nonprivileged Program
$0B	Extended (A32) Nonprivileged Block Transfer
$0D	Extended (A32) Supervisory Data
$0E	Extended (A32) Supervisory Program
$0F	Extended (A32) Supervisory Block Transfer
$29	Short I/O (A16) Nonprivileged Data
$2D	Short I/O (A16) Supervisory Data
$39	Standard (A24) Nonprivileged Data
$3A	Standard (A24) Nonprivileged Program
$3B	Standard (A24) Nonprivileged Block Transfer
$3D	Standard (A24) Supervisory Data
$3E	Standard (A24) Supervisory Program
$3F	Standard (A24) Supervisory Block Transfer

The power up or reset default value for the SZ_AM register is $B9 indicating standard A24 nonprivileged VMEbus addressing using 16-bit transfers.

4.2.12 Self-Test Procedure Register (ST_PROC)

This register determines the set of self-test procedures that the CPU executes when the SLFTST bit in the CR0 register is set to one.

Table 4.2.12-1. Self-Test Procedure Register Bit Map

Global Offset $11 ST_PROC (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
RESERVED	0	TEST_DMA_R	TEST_DMA_W	TEST_BERR	TEST_UART	TEST_TIMER	TEST_VINT

M6016/T4.2.12-1

The following table is a brief summary of the tests. See Section 4.6 for details on self-testing.

Bit	Code	Description
Bit 5	TEST_DMA_R	VMEbus DMA read (bus --> board) test
Bit 4	TEST_DMA_W	VMEbus DMA write (board --> bus) test
Bit 3	TEST_BERR	Self-access BERR test
Bit 2	TEST_UART	UART test (local loopback)
Bit 1	TEST_TIMER	Timer test
Bit 0	TEST_VINT	VMEbus interrupt test

4.2.13 Error Interrupt Control Register (ER_MSK)

The Error Interrupt Control Register is a read/write byte. Bits not mentioned below must remain set to zero (0).

Table 4.2.13-1. Error Interrupt Control Register Bit Map

Global Offset $12 ER_MSK (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
REGBUSY_MSK	RFNZ_MSK	ST_FAIL_MSK	HDWE_ERR_MSK	BERR_MASK	ER_LEVEL		

M6016/T4.2.13-1

Bit 07: Register Busy Transition to Zero (REGBUSY_MSK) – When this bit is set to one (1), a global interrupt will occur whenever data written to the CR0, CR1, GO, TX, or the RX registers has been interpreted by the firmware of the VMIVME-6016. This interrupt may be used as a handshake method in place of setting and polling the REGBUSY bit of the GST register.

Bit 06: Reserved Field Nonzero (RFNZ_MSK) – When this bit is set to one (1), a Reserved Field set to nonzero (RFNZ status) can trigger this interrupt.

Bit 05: Self-Test Failed (ST_FAIL_MSK) – When this bit is set to one (1), a self-test failure (STFAIL status) can trigger this interrupt.

Bit 04: Hardware Error (HDWE_ERR_MSK) – When this bit is set to one (1), a hardware error (HDWEERR status) can trigger this interrupt.

Bit 03: Bus Error (BERR_MSK) – When this bit is set to one (1), a VMEbus error (BERR status) caused by a transfer operation initiated by this board can trigger this interrupt.

Bits 02-00: VMEbus Interrupt Level (ER_LEVEL) – These three bits set the level of the error interrupt from 1-7. If all three bits are clear, the interrupt is effectively disabled since there is no VMEbus level 0 interrupt.

4.2.14 Error Interrupt Vector Register (ER_VEC)

The Error Interrupt Vector register is a byte-wide read/write interrupt vector register. This register should be initialized to hold a standard VMEbus interrupt vector.

Table 4.2.14-1. Error Interrupt Vector Register Bit Map

Global Offset $13 ER_VEC (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
VECTOR							

M6016/T4.2.14-1

4.2.15 Buffer Base Register (BUFBASE)

The Buffer Base register is a read/write longword. When the VMIVME-6016 is in master mode (MASTER bit in CR0), this register contains the VMEbus address for all the user buffers. The contents of the Master Size and Address Modifier Register (SZ_AM) are used to determine access type when addressing the buffers. Note that during VMEbus standard (A24) addressing, the most significant byte of BUFBASE is ignored.

When the VMIVME-6016 is in slave mode, BUFBASE determines the standard or extended VMEbus base address of the on-board memory block used for user buffers. The SLV_32 bit in CR0 determines whether the VMIVME-6016 responds to standard (A24) or extended (A32) addressing. The slave mode base address must be a multiple of a number that depends on the amount of memory populated:

Total Memory	User Memory	Slave Base Multiplier
256 Kbytes	128 Kbytes	256 Kbytes ($40000)
512 Kbytes	256 Kbytes	512 Kbytes ($80000)
1 Mbyte	512 Kbytes	1 Mbyte ($100000)
2 Mbytes	1 Mbyte	2 Mbytes ($200000)

4.2.16 Global Status Register (GST)

The Global Status register is a 16-bit register that contains bits that reflect VMIVME-6016 status for conditions independent of channel number. Bits not mentioned below must remain set to zero (0). VMIC recommends that this register be modified using indivisible RMW operations. See Section 4.4.9 for details. To modify just the REGBUSY bit, a byte write can be performed to offset $19 instead of using an RMW operation.

Table 4.2.16-1. Global Status Register Bit Map

Global Offset $18 GST (Read/Write)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
ANYERR	GLBILL	STBUSY	SCON	RFNZ	STFAIL	HDWEERR	BERR
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
REGBUSY	0	0	0	0	0	0	0

M6016/T4.2.16-1

Bit 15: Any Error (ANYERR) -- This bit is set to one (1) whenever one or more of the following bits are set:

Bit	Description
GLBILL	Illegal Global Register Content
RFNZ	Reserved Field Nonzero
STFAIL	Self-Test Error
HDWEERR	Hardware Error
BERR	BERR* on VMEbus master cycle

Bit 14: Global Register Illegal (GLBILL) – This bit is set to one (1) whenever the VMIVME-6016 detects that the content of a global register violates the restrictions on it.

Bit 13: Self-Test Busy (STBUSY) – This bit is set when the VMIVME-6016 is busy running one of the tests controlled by the Run Self-Test bit (SLFTST) of the CR0 register. After initializing these tests, the host program should wait for this bit to clear before testing the Self-Test Failed bit (STFAIL) in this register to determine the success of the self-tests.

Bit 12: VMEbus System Controller (SCON) – This bit is set to one (1) if the VMIVME-6016 board is jumper-configured as a VMEbus slot 1 controller. See Section 5.3.2 for details.

Bit 11: Reserved Field Nonzero (RFNZ) – This bit is set to one (1) if any reserved bits in any VMIVME-6016 registers are nonzero. If this bit is set, the Any Error bit (bit 15) will also be set. Note that unused register space, which includes space or fields not mentioned in Sections 4.2 and 4.3, is also reserved and must be completely clear. All reserved bits are clear by default after power-up or reset.

Bit 10: Self-Test Failed (STFAIL) – This bit is set to one (1) if any of the self-tests requested by the Run Self-Test bit (SLFTST) of the CR0 Register have failed. This bit is not valid if the Self-Test Busy bit is set. If this bit is set, the Any Error bit (ANYERR) will also be set.

Bit 09: Hardware Error (HDWEERR) – If this bit is set to one (1), some local hardware error other than a VMEbus Error (BERR) has occurred. If this bit is set, the Any Error bit (ANYERR) will be set as well.

Bit 08: VMEbus Error (BERR) – This bit is set if the VMIVME-6016 initiates an operation that results in a bus error. Note that the VMIVME-6016 has a programmable bus timer if it is configured as the system controller (see Section 5.3.2 for details) and has only a fixed-interval bus timer (about 50 microseconds) otherwise. This bit will also be set, however, if a VMIVME-6016 DMA transfer points to the VMIVME-6016 board (a self access). If this bit is set, the Any Error bit (ANYERR) will be set as well.

Bit 07: Register Busy (REGBUSY) – Any write operation into any area of the CR0, CR1, GO, TX, or RX registers causes an interrupt to the firmware on the VMIVME-6016. Until the VMIVME-6016 firmware interprets the new data, the contents of these registers may not reflect the expected results. Register Busy provides a handshake method that allows the host computer software to ensure that the register contents have taken effect. The following code fragment illustrates the use of REGBUSY.

```
/*
* ....................................................
*  Sample code fragment to set a GO bit.
*  GSTLO refers to lower byte of GST at offset $19.
* ....................................................
*/

#define REGBSY 0X80
while((v6016->GSTLO & REGBSY) != 0);     /* let previous finish  */
v6016->GSTLO |= REGBSY;  /* Avoid races by presetting REGBUSY   */
v6016->GO |= 0x400;     /* Set a GO bit */
```

The following code fragment may be used when reading the registers:

```
/*
* ....................................................
* Sample code fragment to read the CR0 register and
* print a message if the LED is emitting light
* ....................................................
*/

while((v6016->GSTLO & REGBSY)!= 0);      /*let previous finish    */
if (v6016->CR0 & LEDN == 0) puts ( "The LED is on" );
```

NOTE:

THE ABOVE HANDSHAKE APPLIES ONLY TO THE CR0, CR1, GO, TX, AND RX REGISTERS; THE REMAINING VMIVME-6016 REGISTERS MAY BE WRITTEN OR READ WITHOUT WAITING FOR REGBUSY TO BE ZERO.

The REGBUSY_MSK interrupt (see Section 4.2.13) may be used to get notification, rather than by polling the GST. See also Section 4.4.8.

4.2.17 Master Granularity (MAS_GRN)

This register is meaningful only in master mode (user buffers off-board). If the number here is zero (0), it represents 256. The user buffer's TX and RX are each allocated this value times 4096 bytes in VMEbus space. Total buffer space is MAS_GRN times 4096 times 32 buffers. The value of SZ_UBUF for each channel determines how much of the buffer space allocated for the channel is actually used.

Table 4.2.17-1. Master Granularity Register Bit Map

Global Offset $1A MAS_GRN (Read/Write)							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
VALUE							

M6016/T4.2.17-1

4.3 CHANNEL CONTROL BLOCKS

Each of the 16 channels of the VMIVME-6016 has a 14-byte Channel Control Block associated with it. These CCBs follow the Global Registers described in Section 4.2, in ascending order, from channel 0 through channel 15. Table 4.3.1 shows the structure of one of the CCBs, followed by its description. Each CCB has the same mapping. The CCB registers are offsets from the base offset of the associated CCB. The Global Registers and the CCBs together take up exactly 256 bytes of short I/O space.

Table 4.3-1. VMIVME-6016 Channel Control Block Map

CCB Offset	Name	Meaning
$00	CST	Channel Status
$02	CH_MSK	Channel Interrupt Mask
$03	CH_VEC	Channel Interrupt Vector
$04	EOB	End-of-block code
$05	XOFF	Flow control XOFF code
$06	XON	Flow control XON code
$07	BRK_DUR	BREAK duration in ms +0/-1
$08	SZ_RING	Internal Ring size
$09	LO_RING	Internal Ring low water mark
$0A	HI_RING	Internal Ring high water mark
$0B	CH_CON1	Channel Control Byte 1
$0C	CH_CON2	Channel Control Byte 2
$0D	SZ_UBUF	User Buffer Size

M6016/T4.3-1

The registers in a CCB are valid only for the particular channel with which the CCB is associated. The VMIVME-6016 can modify the Channel Status (CST) registers at any time. It does not modify any other register in any CCB, except to set up default conditions at power up or hard reset time.

The user should not modify any register in a CCB while its channel is active (associated GO bit set to one).

4.3.1 Channel Status Register (CST)

This register contains all the status bits peculiar to its channel. The user should take care with setting or clearing bits in this register, as the VMIVME-6016 may set or clear bits here at its convenience. This register should be modified with an indivisible RMW operation. See Section 4.4.9 for details.

Table 4.3.1-1. Channel Status Register Bit Map

CCB Offset $00 CST							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
DCD_COS	CTS_COS	DCD	CTS	INTBSY	ILL	URCV_RDY	UTX_RDY
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
BRK_RCVD	FE	PE	OVERRUN	TXEMT	TXRDY	FFUL	RXRDY

M6016/T4.3.1-1

Bit 15: DCD (Data-Carrier-Detect) Changed State (DCD_COS) – A one (1) in this bit indicates that the UART hardware has seen a change of state on the incoming DCD signal.

Bit 14: CTS (Clear-To-Send) Changed State (CTS_COS) – A one (1) in this bit indicates that the UART hardware has seen a change of state on the incoming CTS signal.

Bit 13: DCD State (DCD) – This bit reflects the current state of the incoming DCD line. A one represents an RS-232 space (positive voltage), and a zero represents an RS-232 mark (negative voltage).

Bit 12: CTS State (CTS) – This bit reflects the current state of the incoming CTS line. A one (1) represents an RS-232 space (positive voltage), and a zero (0) represents an RS-232 mark (negative voltage).

Bit 11: Host Interrupt Routine Busy (INTBSY) – Whenever the VMIVME-6016 issues a channel interrupt to the host, it also sets the INTBSY bit. The VMIVME-6016 will not issue another interrupt for ANY channel until the host resets INTBSY. These actions prevent the host from being overrun by channel interrupts. The VMIVME-6016 will not forget to issue requested interrupts, but when it eventually gets permission to interrupt the host because the host reset INTBSY, the CST may reflect several reasons for the interrupt. The host should examine the CST (or several CSTs, if several channels share the same interrupt vector) for all possible reasons for a channel interrupt before resetting INTBSY (or the several INTBSYs) and allowing more interrupts.

Bit 10: Illegal Code (ILL) – A one (1) in this bit indicates that one or more fields in the CCB contained an illegal value when the GO bit for this channel was set to a one (1).

Bit 09: User Receive Buffer Has Data (URCV_RDY) – If this bit is a one (1), the user buffer has data in it ready for the user to take. After taking the data, the user should set the RX register bit for the channel to inform the VMIVME-6016 (unless the channel is configured for ring buffers on-board), which will then load more data into the user buffer. The user must reset URCV_RDY before either taking the data or setting the appropriate RX bit. URCV_RDY sets for one of two reasons: the EOB character (see Section 4.3.4) has been received and stored, or input has timed out (see Section 4.3.12).

Bit 08: User Transmit Buffer Can Take Data (UTX_RDY) – If this bit is a one (1), the user can write new data to the transmit buffer. To make the VMIVME-6016 transmit the new data, the user must set the appropriate bit in the TX register, whereupon the VMIVME-6016 accepts the data and begins transmitting it. The user must reset UTX_RDY before setting the appropriate TX bit. Note that if the user buffer is a ring, the user can put new data into it at any time that the pointers are such that there is enough space in the buffer; the VMIVME-6016 will not recognize the presence of the new data, however, unless it is still busy transmitting, or until the user sets the TX bit.

The following eight CST bits are a ones copy of the UART status register contents for the channel, meaning that once one of these bits sets, the VMIVME-6016 will never clear it. The host must reset these bits. They are updated at the convenience of the on-board CPU, and so will lag the actual UART status when data is being transmitted and/or received.

Bit 07: BREAK received (BRK_RCVD) – This bit, when a one (1), indicates that an RS-232 space level has been received on the RXD line for more bit times than make up one character.

Bit 06: Framing Error (FE) – This bit, when a one (1), indicates that a stop bit was not detected at the expected time when the UART received a character. The stop bit check is made in the middle of the first stop bit position.

Bit 05: Parity Error (PE) – A one (1) in this bit means that an incoming character had incorrect parity. This bit is never set if “none” is selected in PAR_TYP.

Bit 04: Receiver Overrun (OVERRUN) – A one (1) in this bit indicates that one or more incoming characters have been lost. This should not happen unless there is no flow control, or flow control has failed, or a throughput restriction has been violated.

Bit 03: Transmitter Empty (TXEMT) – A one (1) in this bit indicates that the transmitter has run out of data to send. This condition occurs when there is no data in the transmit buffer or the UTX_RDY bit has not been reset by the user.

Bit 02: Transmitter Ready (TXRDY) – A one (1) in this bit means that the UART hardware can take data into its holding register. The transmitter may or may not be empty.

Bit 01: Receiver FIFO full (FFUL) – A one (1) in this bit means that the UART hardware receiver FIFO has filled, that is, the on-board CPU can read three or possibly four characters from the UART.

Bit 00: Receiver Ready (RXRDY) – A one (1) in this bit indicates that the UART has at least one character in its hardware receiver FIFO.

4.3.2 Channel Interrupt Mask Register (CH_MSK)

This register controls the reasons for sending a channel specific interrupt to the VMEbus. The VMIVME-6016 issues a channel interrupt whenever a mask-selected event occurs that can also set the corresponding CST status bit. It does not matter whether or not the status is already set. The host is responsible for resetting these status bits.

Table 4.3.2-1. Channel Interrupt Mask Register Bit Map

CCB Offset $02 CH_MSK							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
COS_MSK	ERR_MSK	BREAK_MSK	RCV_MSK	TX_MSK	CH_LEVEL		

M6016/T4.3.2-1

Bit 07: CTS or DCD changed state (COS_MSK) – When set to one (1), this bit sends an interrupt whenever either the CTS or DCD incoming lines change state.

Bit 06: FE, PE, OVERRUN, or illegal code (ERR_MSK) – When set to one (1), this bit sends an interrupt whenever one or more of these errors are detected.

Bit 05: BREAK received (BREAK_MSK) – When set to one (1), this bit sends an interrupt whenever the receive function detects an incoming BREAK.

Bit 04: User receive buffer has data (RCV_MSK) – When set to one (1), this bit sends an interrupt whenever the VMIVME-6016 deposits data in the user receive buffer and sets URCV_RDY.

Bit 03: User transmit buffer can take data (TX_MSK) – When set to one (1), this bit sends an interrupt whenever the user transmit buffer goes from the full state to not full (ring), or can take data (linear) and sets UTX_RDY.

Bits 02-00: Channel Interrupt Level (CH_LEVEL) – This field sets the level of the VMEbus interrupt for channel interrupts. A zero (0) here disables the channel interrupts.

4.3.3 Channel Interrupt Vector Register (CH_VEC)

This register holds the vector presented to the VMEbus during an IACK cycle acknowledging a channel interrupt.

Table 4.3.3-1. Channel Interrupt Vector Register Bit Map

CCB Offset $03 CH_VEC							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
VECTOR							

M6016/T4.3.3-1

4.3.4 End-of-Block Code Register (EOB)

If this register is nonzero, reception of a character with its value causes URX_RDY to set (and send a VMEbus interrupt if RCV_MSK is enabled) even if the user receive buffer is not full.

Table 4.3.4-1. End-of-Block Code Register Bit Map

CCB Offset $04 EOB							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
VALUE							

M6016/T4.3.4-1

4.3.5 Flow Control XOFF Code Register (XOFF)

This register holds the code, if received, to be used to stop the transmitter, provided FLOWCON is set either to XON/XOFF or to ANY/XOFF.

Table 4.3.5-1. Flow Control XOFF Code Register

CCB Offset $05 XOFF							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
CODE							

M6016/T4.3.5-1

4.3.6 Flow Control XON Code Register (XON)

This register holds the code which, if received, restarts the transmitter, provided FLOWCON is set to XON/XOFF.

Table 4.3.6-1. Flow Control XON Code Register Bit Map

CCB Offset $06 XON							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
CODE							

M6016/T4.3.6-1

4.3.7 BREAK Duration Register (BRK_DUR)

This register determines the duration of transmitted BREAK signals. A BREAK is sent whenever the BREAK register bit associated with the channel is set to one (1), TXBRK is set to one (1), and lasts the number of milliseconds set in BRK_DUR with a maximum of 255 ms possible. When the BREAK ends, the BREAK bit resets to zero (0).

Table 4.3.7-1. BREAK Duration Register Bit Map

CCB Offset $07 BRK_DUR							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
DURATION IN MILLISECONDS							

M6016/T4.3.7-1

4.3.8 Internal Ring Size Register (SZ_RING)

The internal (not the user) ring buffer is SZ_RING blocks of 128 bytes long. A value of zero is not allowed and causes the ILL bit in the CST to be set to one (1).

Table 4.3.8-1. Internal Ring Size Register Bit Map

CCB Offset $08 SZ_RING							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
NUMBER OF BLOCKS							

M6016/T4.3.8-1

The maximum allowable value of SZ_RING depends on the amount of memory available on-board:

Total Memory	User Memory	Max SZ_RING (decimal)
256 Kbytes	128 Kbytes	8
512 Kbytes	256 Kbytes	40
1 Mbyte	512 Kbytes	104
2 Mbytes	1 Mbyte	232

4.3.9 Internal Ring Low Water Mark Register (LO_RING)

This register sets the number of bytes stored in the input internal ring below which flow control restarts input. See Section 4.2.11.

Table 4.3.9-1. Internal Ring Low Water Mark Register Bit Map

CCB Offset $09 LO_RING							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
NUMBER OF BYTES							

M6016/T4.3.9-1

4.3.10 Internal Ring High Water Mark Register (HI_RING)

This register sets the number of bytes not in use in the input internal ring below which flow control suspends input. See FLOWCON, below.

Table 4.3.10-1. Internal Ring High Water Mark Register Bit Map

CCB Offset $0A HI_RING							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
NUMBER OF BYTES							

M6016/T4.3.10-1

4.3.11 Channel Control Byte 1 Register (CH_CON1)

This register controls the kind of user buffer (linear or ring), flow control strategy, and the bit rate for the channel.

Table 4.3.11-1. Channel Control Byte 1 Register Bit Map

CCB Offset $0B CH_CON1							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	BIt 02	Bit 01	Bit 00
UB_RING	FLOWCON		BAUD				

M6016/T4.3.11-1

Bit 07: User Buffer Type (UB_RING) –This bit, if set to one (1), implements a ring mode user buffer. In ring mode, the first two bytes (or four bytes when using Master mode) of the buffer are an input pointer and the next two bytes (or four bytes in Master mode) are an output pointer as offsets from the start of the actual buffer, which begins immediately after the output pointer. The terms "input pointer" and "output pointer" refer to input and output from the point of view of the associated buffer, not with respect to the host or the VMIVME-6016. The user examines both pointers to determine how full the ring buffer is. The buffer is empty if the pointers are equal. The buffer is full if the input pointer is one behind the output pointer. For a receive buffer, the user extracts data from the buffer and then advances the output pointer. For a transmit buffer, the user deposits data in the buffer and then advances the input pointer.

NOTE:

THE USER SHOULD NEVER MODIFY THE OUTPUT POINTER OF A TRANSMIT BUFFER OR THE INPUT POINTER OF A RECEIVE BUFFER; THESE ARE MAINTAINED BY THE VMIVME-6016. SERIOUS MALFUNCTIONS MAY OCCUR IF EITHER OF THESE POINTERS IS CHANGED BY THE USER/HOST; EVEN A SYSTEM-LEVEL CRASH COULD OCCUR.

If the user has selected both ring buffer and slave mode (buffers on-board), there is NO internal ring buffer; input and output go to and from the user buffers directly. (The internal buffer is eliminated to improve throughput.) Consequently for this case, the RX bit has no function, SZ_RING is not used, and the HI_RING and LO_RING quantities refer to the user buffers.

If UB_RING is set to zero, the user buffer is a linear buffer (no pointers). In this case, the user examines the URCV_RDY or UTX_RDY bits in the CST register and reads or writes the entire buffer contents accordingly. The first two bytes (or four bytes when using Master mode) of a linear buffer form a 16-bit (or 32-bit in Master mode) count of the number of meaningful characters in the buffer; the actual buffer of characters follows this counter.

The VMIVME-6016 implements the channels with SCC2698B octal UART devices; the channels are in some way controlled in pairs. The pairs are: 0,1 2,3 4,5 6,7 etc. If the user buffers are on-board (Slave mode), UB_RING in the members of a pair must be the same. For example, if all the user buffers are on-board and if channel 12 is configured for ring buffers (UB_RING=1), then channel 13 must be configured for ring buffers. No such restriction applies if the user buffers are off-board (Master mode).

Bits 06-05: Flow Control (FLOWCON) – This field allows the VMIVME-6016 to act to prevent loss of data due to data rate restrictions at either end of a channel. The transmit function reacts to XOFF or the negation of CTS by stopping transmission until the receiver detects an XON (or any character if so selected), or CTS reasserts. The receiver function sends an XOFF or negates RTS when the internal receive ring exceeds the high water mark (HI_RING), and sends XON or asserts RTS when the internal receive ring goes below the low water mark (LO_RING). The user can optimize flow control by adjusting the internal ring size (SZ_RING) and the high and low water marks. The codes for FLOWCON are:

Bit 06	Bit 05	Flow Control
0	0	None
0	1	XON/XOFF
1	0	Any/XOFF
1	1	RTS/CTS

Bits 04-00: Baud rate for both TX and RX (BAUD) – This field controls the bit rate for both transmitted and received data. The coding for the BAUD field is:

Bit 04	Bit 03	Bit 02	Bit 01	Bit 00	Baud Rate
0	0	0	0	0	50
0	0	0	0	1	110
0	0	0	1	0	134.5
0	0	0	1	1	200
0	0	1	0	0	300
0	0	1	0	1	600
0	0	1	1	0	1200
0	0	1	1	1	1050
0	1	0	0	0	2400
0	1	0	0	1	4800
0	1	0	1	0	7200
0	1	0	1	1	9600
0	1	1	0	0	38400
1	0	0	0	0	75
1	0	0	0	1	110
1	0	0	1	0	38400
1	0	0	1	1	150
1	0	1	0	0	300
1	0	1	0	1	600
1	0	1	1	0	1200
1	0	1	1	1	2000
1	1	0	0	0	2400
1	1	0	0	1	4800
1	1	0	1	0	1800
1	1	0	1	1	9600
1	1	1	0	0	19200

The VMIVME-6016 implements the channels with SCC2698B octal UART devices; the channels are in some ways controlled in pairs. Channels 0 and 1 have the most significant bit of their BAUD fields in common, for example. Therefore, the BAUD fields for an even-odd pair of channels must have the SAME most significant bit value.

4.3.12 Channel Control Byte 2 Register (CH_CON2)

This register controls the parity and number of bits per character of both transmitted and received data.

Table 4.3.12-1. Channel Control Byte 2 Register Bit Map

CCB Offset $0C CH_CON2							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
INP_TIMEOUT		TWO_STOP	PAR_TYP		PAR_ODD	SZ_CHAR	

M6016/T4.3.12-1

Bits 07-06: Input Timeout Mode (INP_TIMEOUT) – This field selects the input timeout method for setting URCV_RDY and the corresponding host interrupt. If incoming characters stop arriving for the chosen timeout period, URCV_RDY will set and the selected host interrupt will occur, just as if the EOB character had been received. (URCV_RDY may also be set by the reception of an EOB character; see Section 4.3.4.) The timeout method coding is:

Bit 07	Bit 06	Timeout Mode
0	0	none
0	1	3 char times or break duration, whichever is greater
1	0	1/2 second
1	1	1 second

Bit 05: Two Stop Bits (TWO_STOP) – This bit, if set to a one (1), will cause all transmitted characters for the channel to end with a two bit-time stop (RS-232 mark state). If this bit is a zero, the stop will be one bit-time long.

Bits 04-03: Transmit/Receive Parity Type (PAR_TYP) – This field determines whether or not a parity bit exists in both transmitted and received data, and if so, its sense. The code for PAR_TYP is:

Bit 04	Bit 03	Parity
0	0	Odd or Even, use PAR_ODD
0	1	Force to PAR_ODD state
1	0	No parity bit

Bit 02: Parity Sense is Odd (PAR_ODD) – This bit determines the sense of the parity bit, if used. If odd or even parity is selected, the parity is odd if PAR_ODD is a one, and even if it is a zero. If forced parity is selected, PAR_ODD is the value of the parity bit.

Bits 01-00: Bits/char not including parity bit (SZ_CHAR) – This field determines the number of bits, exclusive of a possible parity bit, per transmitted or received character. The coding is:

Bit 01	Bit 00	Bits/Char
0	0	5
0	1	6
1	0	7
1	1	8

4.3.13 User Buffer Size Register (SZ_UBUF)

This register determines the size of the user buffer in bytes.

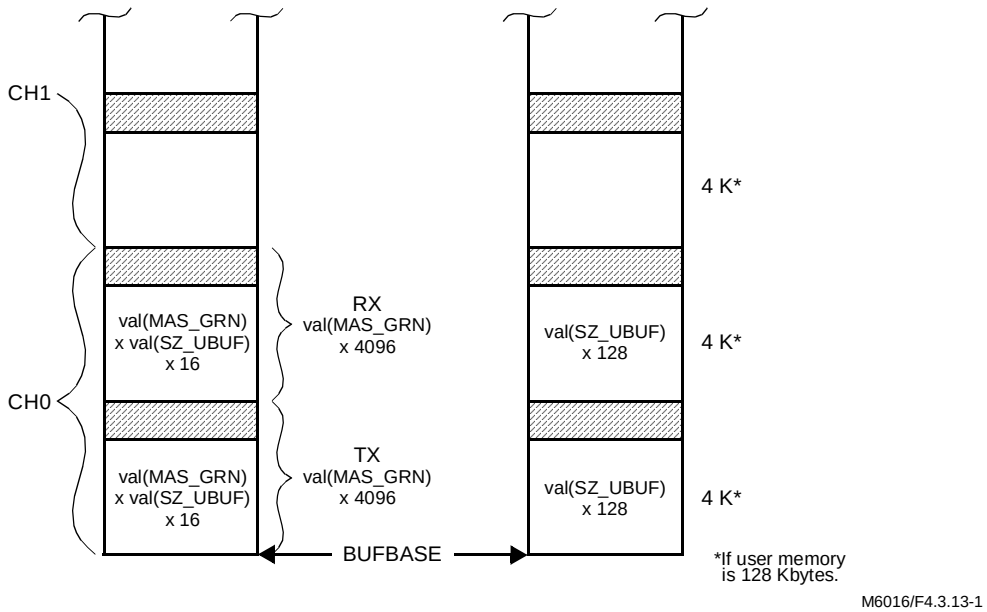
Table 4.3.13-1. User Buffer Size Register Bit Map

CCB Offset $0D SZ_UBUF							
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00

M6016/T4.3.13-1

In master mode, the size is 16 times the value in this register times the value represented by the content of the global register MAS_GRN. If the SZ_UBUF value is zero (meaning 256), the buffer size is 4096 times the value in the MAS_GRN register (zero meaning 256). Buffers will be allocated starting at BUFBASE for the TX buffer and RX buffer of Channel 0. Channel 1 will be at BUFBASE plus MAS_GRN times 4096 continued through Channel 15 (see Figure 4.3.13-1). In slave mode, the size is 128 times the value in the register (zero meaning 256), and the maximum allowable value of SZ_UBUF depends on the amount of memory available on-board:

User Memory	Max SZ_UBUF (decimal)	Buffer Size
128 Kbytes	32	4096
256 Kbytes	64	8192
512 Kbytes	128	16384
1 Mbyte	0	32768



M6016/F4.3.13-1

Figure 4.3.13-1. Buffers

4.4 GENERAL PROGRAMMING DETAILS

4.4.1 Introduction

The VMIVME-6016 can perform data transfers automatically for up to 16 channels. These may be any mix of baud rates, byte formats, etc., but must all be for the same VMEbus address space. The user must load the VMIVME-6016 on-board registers with information for configuring the channels. After the registers have been set up, the user must set the GO register bits for the appropriate channels to start the VMIVME-6016. Table 4.4.1-1 (at the end of Section 4.4) is an alphabetized glossary of the names of the various registers and their bit fields. Sections 4.2 and 4.3 describe the registers in detail. As an example of how to use the glossary, suppose that the user wants to set the channel interrupt level for channel 3 to level 5. CH_LEVEL in the glossary shows that it is the rightmost three bits of the CH_MSK register (mask is $07). The CH_MSK entry shows that it is at offset $02 within a CCB (Channel Control Block). Because the Global Registers take up $20 bytes, and each CCB takes up $0E bytes, the CCB for channel three begins at 20 + (3 x 0E) = $4A. Therefore, the user should deposit 101 binary (5) in the rightmost three bits of short I/O $XX4C, where XX is set by jumpers as shown in Section 5.3.1.

NOTE:

IT IS IMPORTANT TO REALIZE THAT ANY HARDWARE ERROR OR VMEbus ERROR INTERRUPT THAT COMES FROM THE VMIVME-6016 MEANS THAT VMIVME-6016 ACTIVITY HAS STOPPED. THE CPU MUST TAKE ACTION TO ALLOW ACTIVITY TO CONTINUE.

4.4.2 Channel Shutdown

The host computer should shut down activity on a channel before modifying any of the CCB registers for the channel. If any global registers are to be modified, except for the GST register, all channels should be shut down. The host shuts down a channel by setting the REGBUSY bit in the GST register, resetting the GO register bit for that channel, and then waiting for the REGBUSY bit to go to zero (see Section 4.2.16).

4.4.3 Global Setup for Board Operation

The host should set up ER_MSK and ER_VEC before setting any other registers. This allows the VMIVME-6016 to assert an error interrupt in response to register set-up errors. For example, if the host sets some reserved field or register nonzero, the RFNZ status bit would set, and ER_MSK and ER_VEC would (if so masked) allow the VMIVME-6016 to interrupt the host.

The host must set the MASTER bit in CR0 to determine whether the VMIVME-6016 operates in master mode (user buffers off-board) or in slave mode (user buffers on-board). The meaning of some other registers and fields depend on the setting of this bit.

If the VMIVME-6016 will be in master mode (user buffers off-board), the host must set up SZ_AM, CR1, and CR2. SZ_AM determines the width (VMESIZE), and the AM bits (ADDRMOD) of master VMEbus transfers. CR2 determines the bus request level (BRLEVEL), the release mode (RELMOD), and the fairness timeout (FAIR_TO) for master transfers. In the slave mode, these two registers do not matter. The BERR_TO field in CR1 determines VMEbus cycle timeout, or can disable the timer. The ARBMOD bit in CR1 is meaningful only if the VMIVME-6016 is the VMEbus System Controller in VMEbus slot 1 (see Section 5.3.2).

The BUFBASE register serves two purposes. In the master mode, it points to the VMEbus address of the user transmit buffer for channel 0. (The other 31 user buffers follow that one, alternating transmit and receive buffers in ascending channel order.) In slave mode, BUFBASE is the programmable base VMEbus address of the on-board user buffers, which are arranged in the on-board memory the same as they are for the master mode.

Next, the host should set up the rest of CR0. In the slave mode, SLV_32 controls whether the VMIVME-6016 responds to standard or extended addressing, SUPONLY controls whether the board responds only to supervisory or to both supervisory and nonprivileged addressing, and SLV_OK allows the board to respond as a VMEbus slave for user buffer space. Note that SLV_OK should not be set until SUPONLY, SLV_32, and BUFBASE are set up correctly, or the VMIVME-6016 might respond in an unintended way.

To complete the global setup, the host must ensure that all bits in the BREAK and ST_PROC registers are zeros.

4.4.4 Channel Setup

For each channel to be run, the host must set up the associated CCB. The first two bytes of the CCB comprise the CST register, which automatically initializes when the host starts the channel.

The host should set the CH_MSK and CH_VEC registers to let the appropriate interrupts be serviced by host interrupt service routines.

The EOB, XOFF, XON, BRK_DUR, SZ_RING, LO_RING, and HI_RING registers should be loaded as needed. See the descriptions of these registers in Section 4.3.

The SZ_UBUF register represents the amount of memory actually used from the space allocated to each of the user transmit and receive buffers.

The host must set up the fields in CH_CON1 and CH_CON2 registers, which control bits/character, parity, flow control, bits/second, and buffer architecture (ring or linear). See Section 4.3 for details.

4.4.5 Channel Startup

After initializing code and interrupt service routines, the host may start VMIVME-6016 automatic operation by setting the bit in the GO register that corresponds to the channel to be started. The VMIVME-6016 automatically initializes the CST register for any channel when that channel is started.

Once one or more channels are active, others can be started or stopped without interfering with active channels provided that none of the global registers needs to be modified (other than the GST register).

4.4.6 Channel Operation

If the user buffers are ring buffers, "full" means that the buffer input pointer is one behind the output pointer, and "empty" means that the pointers are equal.

After the GO register bit for a channel is set, the channel begins to receive data and store it in the user receive buffer. The VMIVME-6016 notifies the host that the user buffer has data available by setting a channel status bit and possibly interrupting the host. The host, after taking the data, advances the receive ring buffer output pointer or resets the character count in the receive linear buffer, and tells the VMIVME-6016 to resume storing into the user receive buffer by setting the appropriate bit in the RX register. Note that the host must set the RX bit to tell the VMIVME-6016 to load the receive user buffer the first time. The VMIVME-6016 will reset the RX bit, set URCV_RDY, and possibly interrupt the host when either an EOB character arrives or input timeout occurs. If either the user receive buffer fills up or the hardware FIFO in the UART overflows, the VMIVME-6016 will sets OVERRUN in the CST register for the channel, and will possibly interrupt the host. If the VMIVME-6016 is in slave mode and the user buffers are ring buffers, there is no internal ring buffer, so the host does not need to tell the VMIVME-6016 to transfer data to the user buffer; here the RX bit is meaningless.

After the host loads data to be transmitted into the user transmit buffer, and advances the transmit ring buffer input pointer or sets the character count into the transmit linear buffer, it tells the VMIVME-6016 to take the data and begin transmitting it by setting the appropriate bit in the TX register. It notifies the host that the data is all accepted (transmit buffer available for new data) by setting a channel status bit and possibly interrupting the host. The VMIVME-6016 also automatically resets the TX register bit when it shuts down the transmitter for lack of data to transmit. If the VMIVME-6016 is in slave mode and the user buffers are ring buffers, there is no internal ring buffer. In this case, the host may insert new data into the user ring buffer while the VMIVME-6016 is transmitting, and the new data will be transmitted. If this is done, the host UTX_RDY interrupt service should check the transmit buffer for data, and take steps to be sure it gets sent out; the VMIVME-6016 may have stopped transmitting just as the new data was given to it.

4.4.7 Restarting After an Error Interrupt

Interrupts from a channel (CH_MSK and CH_VEC) do NOT stop activity, but, as noted above, a global error interrupt WILL stop ALL channel activity. (The REGBUSY_MSK interrupt, although global, is not an error and so does not stop channel activity.) In the latter case, the host must restart ALL the channels that should be active. Incoming characters may be lost as a result of a global error interrupt and without receiving overrun errors.

4.4.8 Performance Considerations

The registers and on-board buffers reside in memory used by the on-board CPU for code, stack space, local variables, internal ring buffers, and so on. When the host accesses the registers or on-board buffers, the on-board CPU has to wait for such accesses to complete. This loss of time negatively impacts performance. Therefore, accesses by the host should be restricted to the essentials. For example, the host should not poll status; it should instead make use of interrupts, and examine status when they occur. It also helps to transfer a number of characters to or from the buffers and update pointers or counters after the transfer, rather than updating pointers on a character-by-character basis.

4.4.9 Usage Notes on VMIVME-6016 Registers

The VMIVME-6016's registers are actually on-board memory locations that are shared by the VMIVME-6016's processor and the VMEbus. Because of this arrangement, some of the VMIVME-6016's registers should only be modified using an indivisible VMEbus read modify write instruction. The registers that have the indivisible RMW requirement are: CR0, GO, TX, RX, and CST registers for each channel. For CR0, the requirement for indivisible RMWs can be bypassed if the following cautions are observed:

1) If you set the TXBRK bit, do not modify CR0 until the VMIVME-6016 clears the TXBRK bit.

2) If you set the SLFTST bit, do not modify CR0 until the VMIVME-6016 clears the SLFTST bit.

The need to use RMW cycles for accessing the GO, TX, RX, and CST registers can be seen by looking at the TX register. This register has 16 bits, one for each channel. To begin a transmit on a channel or channels, the user sets the appropriate bits. If the user then wants to do a transmit on some other channel, the user must read the TX register and OR it with a mask for the additional channels. The result is then written back to the VMIVME-6016's TX register. Note: When the VMIVME-6016 has completed a TX for a particular channel, it clears the corresponding bit in the TX register. The RMW requirement stems from the following sequence of events:

1) User writes TX channel setting bits 0, 1, and 2.

2) The VMIVME-6016's on-board CPU starts a transmit on channels 0, 1, and 2.

3) User software reads $0007 from the TX register and stores it in a variable, say TX_REG.

4) Meanwhile, the VMIVME-6016 firmware completes the transmit on channel 2, and clears bit 2 of the TX register. TX register now equals $0003.

5) User software sets bit 3 of TX_REG. TX_REG now equals $000F (it needs to be $000B to avoid a false start on channel 2). User then writes TX_REG to the VMIVME-6016's TX register. By doing so, the user inadvertently starts another transmit on channel 2.

For the example given above, the false start on channel 2 can be avoided by removing step 5 and using an indivisible RMW operation in step 3 that sets bit 3 of the TX register. The Global Status Register (GST) can be treated as two separate registers: high byte and low byte. It is recommended that the upper byte of the GST register be modified using indivisible RMW operations. The low byte does not need indivisible RMWs since REGBUSY is the only bit defined in the low byte.

Table 4.4.1-1. VMIVME-6016 Register and Field Symbols

Symbol	Glob/Chan -or- Reg. Symbol	Size+Offset -or- Bitmask	Description
ADDRMOD	SZ_AM	3F	VMEbus Address Modifier
ANYERR	GST	8000	Any Error (OR of bits marked ERROR)
ARBMOD	CR1	08	SCON VMEbus Arbiter mode: 1=Priority 0=RR
BAUD	CH_CON1	1F	Baud rate for both TX and RX
BERR	GST	0100	BERR on VMEbus master cycle (ERROR)
BERR_MSK	ER_MSK	08	VMEbus BERR mask
BERR_TO	CR1	07	VMEbus BERR timeout if SCON
BRD_ID	(glob)	Byte+00	Board ID
BREAK	(glob)	Word+0C	“BREAK”, ch[15..0]=BREAK[15..0]
BREAK_MSK	CH_MSK	20	BREAK received
BRK_DUR	(chan)	Byte+07	BREAK duration in ms +0/-1
BRK_RCVD	CST	0080	BREAK received
BRLEVEL	CR2	C0	VMEbus Bus Request Level to use
BUFBASE	(glob)	Long+14	Base of buffers (M) or board (S)
CH_CON1	(chan)	Byte+0B	Channel Control Byte 1
CH_CON2	(chan)	Byte+0C	Channel Control Byte 2
CH_LEVEL	CH_MSK	07	Channel Interrupt Level
CH_MSK	(chan)	Byte+02	Channel Interrupt Mask
CH_VEC	(chan)	Byte+03	Channel Interrupt Vector
COS_MSK	CH_MSK	80	CTS or DCD changed state
CR0	(glob)	Byte+04	Command Register 0
CR1	(glob)	Byte+05	Command Register 1
CR2	(glob)	Byte+0E	Command Register 2
CST	(chan)	Word+00	Channel Status
CTS	CST	1000	CTS state
CTS_COS	CST	4000	CTS (Clear-To-Send) changed state
DCD	CST	2000	DCD state
DCD_COS	CST	8000	DCD (Data-Carrier-Detect) changed state
EOB	(chan)	Byte+04	End-of-block code
ERR_MSK	CH_MSK	40	FE, PE, OVERRUN, or illegal code
ER_LEVEL	ER_MSK	07	VMEbus level code
ER_MSK	(glob)	Byte+12	Error interrupt control
ER_VEC	(glob)	Byte+13	Error interrupt vector
FAIR_TO	CR2	0F	Fairness control and timeout
FE	CST	0040	Framing Error

M6016/T4.4.1-1/2

Table 4.4.1-1. VMIVME-6016 Register and Field Symbols (Continued)

Symbol	Glob/Chan -or- Reg. Symbol	Size+Offset -or- Bitmask	Description
FFUL	CST	0002	Receiver FIFO full
FLOWCON	CH_CON1	60	Flow control
GO	(glob)	Word+06	"GO" bits, ch[15..0] = GO[15..0]
GST	(glob)	Word+18	Global Status register
HDWEERR	GST	0200	6016 hardware error (ERROR)
HDWE_ERR_MSK	ER_MSK	10	Hardware Error mask
HI_RING	(chan)	Byte+0A	Internal Ring high water mark
ILL	CST	0400	Illegal BAUD or PAR_TYP, or other param
INP_TIMEOUT	CH_CON2	80	Input Timeout Control
LEDN	CR0	01	1=LED off, 0=LED on
LO_RING	(chan)	Byte+09	Internal Ring low water mark
MASTER	CR0	80	1=Master (buffers off-board), 0=Slave mode
MAS_GRN	(glob)	Byte+1A	Master Granularity
OPTIONS	(glob)	Word+02	ROM version code and hardware options
OVERRUN	CST	0010	Receiver Overrun
PAR_ODD	CH_CON2	04	Parity: 1=odd 0=even
PAR_TYP	CH_CON2	18	Transmit/Receive Parity Type
PE	CST	0020	Parity Error
RCV_MSK	CH_MSK	10	User receive buffer has data
REGBUSY	GST	0080	Write-to-register process busy
REGBSY	GST	80	bit MASIL for lower byte of GST
REGBUSY_MSK	ER_MSK	80	Register Busy Transition to Zero
RELMOD	CR2	30	VMEbus Release Mode
RFNZ	GST	0800	Reserved Field set nonzero (ERROR)
RFNZ_MSK	ER_MSK	40	Reserved-filed-nonzero mask
RST_BRD	CR0	02	1=reset entire board, 0=normal (run)
RX	(glob)	Word+0A	“RX” bits, ch[15..0] = RX[15..0]
RXRDY	CST	0001	Receiver Ready (has a character)
SCON	GST	1000	1 = SCON jumper is installed (slot 1)
SLFTST	CR0	04	1=self-test, 0=normal
SLV_32	CR0	20	1=slave is A32, 0=slave is A24
SLV_OK	CR0	40	1=A24/A32 slave decode OK
STBUSY	GST	2000	Self-Test function busy
STFAIL	GST	0400	6016 self-test error (ERROR)
STFLAG	(glob)	Byte+01	Self-Test flag in ID register
ST_FAIL_MSK	ER_MSK	20	Self-Test error mask
ST_PROC	(glob)	Byte+11	Self-Test procedure register

M6016/T4.4.1-1/3

Table 4.4.1-1. VMIVME-6016 Register and Field Symbols (Continued)

Symbol	Glob/Chan -or- Reg. Symbol	Size+Offset -or- Bitmask	Description
SUPONLY	CR0	10	1=slave is SUP only, 0=slave is SUP and NP
SZ_AM	(glob)	Byte+10	Master's size and AM
SZ_CHAR	CH_CON2	03	Bits/char not including parity bit
SZ_RING	(chan)	Byte+08	Internal Ring size
SZ_UBUF	(chan)	Byte+0D	User Buffer Size
TEST_BERR	ST_PROC	08	Self-access BERR test
TEST_DMA_R	ST_PROC	20	VMEbus DMA read (bus --> board) test
TEST_DMA_W	ST_PROC	10	VMEbus DMA write (board --> bus) test
TEST_TIMER	ST_PROC	02	Timer test
TEST_UART	ST_PROC	04	UART test (local loopback)
TEST_VINT	ST_PROC	01	VMEbus interrupt test
TWO_STOP	CH_CON2	20	Transmit stop bits: NZ=2, Z=1
TX	(glob)	Word+08	“TX” bits, ch[15..0] = TX[15..0]
TXBRK	CR0	08	1=Send BREAK signals, as marked in BREAK
TXEMT	CST	0008	Transmitter Empty
TXRDY	CST	0004	Transmitter Ready (but maybe not empty)
TX_MSK	CH_MSK	08	User transmit buffer needs data
UB_RING	CH_CON1	80	User Buffer type: 1=ring 0=linear
URCV_RDY	CST	0200	User receive buffer has data
UTX_RDY	CST	0100	User transmit buffer can take data
VMESIZE	SZ_AM	C0	VMEbus size, 00 = L, 01 = B, 10 = W
VME_SIZE_16	SZ_AM	80	16-bit size as VMEbus master
VME_SIZE_32	SZ_AM	00	32-bit size as VMEbus master
VME_SIZE_8	SZ_AM	40	8-bit size as VMEbus master
XOFF	(chan)	Byte+05	Flow control XOFF code
XON	(chan)	Byte+06	Flow control XON code

M6016/T4.4.1-1/3

4.5 GENERAL PROGRAMMING EXAMPLES

Here is a simple example of how to use the VMIVME-6016. This program makes channel 1 echo incoming characters to its output, line by line, assuming each line ends with a carriage return ($0D). The header file is shown in Section 4.7.

```
/* File:           6016tst1.c
 * Description:    Echo input to output on one 6016 channel
 * Revision:       XO
 * Date:           21 July 94
 * Where used:     Force-33 CPU and memory in chassis with a VMIVME-6016
 * Property of:    VMIC (VME Microsystems International Inc.)
 */

#define CHANNEL 1           /* 6016 channel to use */
#define BASE_AD 0x6000      /* Short I/O register base */
#define BUF_AD  0x40000     /* Standard (A24) buffer memory base */
#define UBUF_SZ 128         /* User ring size (multiple of 128) */
#define ERVEC   0x81        /* Vector for global error interrupts */
#define ERLEV   4           /* Level for global error interrupts */
#define CHVEC   0x80        /* Vector for channel interrupts */
#define CHLEV   3           /* Level for channel interrupts */

#include <stdio.h>

typedef unsigned long Long;
typedef unsigned short Word;
typedef unsigned char Byte;
#define VME_SHORT_IO 0xFBFF0000
#define VME_STANDARD 0xFB000000

#include "6016hdwe.h"      /* User register definitions */
#include “setclr.h”        /* Header definition for 680x0 RMW instructions*/

void setvec(Long, void (*) (void));
void abort(void);
void err_isr(void);
void chn_isr(void);

Word channel;
Word ubuf_sz;
Byte *regbase;
Byte *chnbase;
Word rxflag;
Word txflag;
Word * txi_index_ptr;
Word * txo_index_ptr;
Word * rxi_index_ptr;
Word * rxo_index_ptr;
Byte * rx_ring;
Byte * tx_ring;
```

```
void
main()
{
   /* Set up base addresses for global registers and CCB */
   channel = CHANNEL;
   ubuf_sz = UBUF_SZ - 4;
   regbase = (Byte *)(VME_SHORT_IO | BASE_AD);   /* Assume a 6016 here */
   chnbase = regbase + GREG_SIZE + CREG_SIZE * channel;

   /* Set up pointers to ring buffers and indexes */
   tx_ring = (Byte *)(VME_STANDARD + BUF_AD + channel * 0x2000);
   rx_ring = tx_ring + 0x1000;
   txi_index_ptr = (Word *)tx_ring;
   txo_index_ptr = txi_index_ptr + 1;
   rxi_index_ptr = (Word *)rx_ring;
   rxo_index_ptr = rxi_index_ptr + 1;

   /* Attach the interrupt service routines */
   setvec((Long)ERVEC, err_isr);      /* Point to error handler */
   setvec((Long)CHVEC, chn_isr);      /* Point to channel handler */

   /* Make sure we are shut down */
   rxflag = 0;                        /* Initialize interrupt flags */
   txflag = 1;
   GSTLO |= REGBSY;                   /* Preset registers busy */
   GO = 0;                            /* Make sure it's dead */
   while (GSTLO & REGBSY);            /* Wait for death */

   /* Global setup */
   GST = 0;                           /* Clean out global status */
   ER_VEC = ERVEC;                    /* Vector for global errors */
   ER_MSK = RFNZ_MSK | HDWE_ERR_MSK | ERLEV; /* Level ERLEV, two reasons */
   BREAK = 0;
   ST_PROC = 0;                       /* No self tests */
   BUFBASE = VME_STANDARD | BUF_AD;   /* User buffers */
   GSTLO |= REGBSY;                   /* Preset registers busy */
   CR0 = SLV_OK | LEDN;               /* Present user buffers, LED off */
   while (GSTLO & REGBSY);            /* Wait for registers not busy */

   /* Channel setup */
   CH_VEC = CHVEC;                    /* Vector for channel interrupts */
   CH_MSK = RCV_MSK | TX_MSK | CHLEV; /* Level CHLEV, xmit done or rcv rdy */
   CST = 0;                           /* Clean out channel status */
   EOB = 0x0D;                        /* EOB is a carriage return */
   XOFF = 0x13;                       /* XOFF is a cntl-S */
   XON = 0x11;                        /* XON is a cntl-Q */
   BRK_DUR = 8;                       /* Eight millisecond BREAK */
    SZ_RING = 1;                      /* Smallest internal ring */
   LO_RING = 32;                      /* Low water is 32 chars */
   HI_RING = 32;                      /* High water is 32 chars */
   CH_CON1 = UB_RING | 0x20 | 0x0B;   /* User ring, XOFF/XON, 9600 bits/sec */
   CH_CON2 = 0x10 | 0x03;             /* 1 stop bit, no parity, 8 bits/char */
   SZ_UBUF = (ubuf_sz + 4) / 128;     /* Set size of user ring */
   /* Enable the channel */
   GSTLO |= REGBSY;                   /* Preset registers busy */
   GO |= (1 << channel);              /* Turn on the channel */
                                      /* If using more than 1 channel, modify
                                      GO REG using indivisible RMW
                                      instructions*/
   while (GSTLO & REGBSY);            /* Wait for registers not busy */
```

```
   /* Copy input to output indefinitely, a "line" at a time */
   while (1) {

      /* Wait for input */
      while (!rxflag);                 /* Wait for EOB input */
      rxflag = 0;

      /* Move the input to the output */
      while (*rxo_index_ptr != *rxi_index_ptr) {
         *(tx_ring + *txi_index_ptr + 4L) = *(rx_ring + *rxo_index_ptr + 4L);
         if (*rxo_index_ptr >= ubuf_sz)
            *rxo_index_ptr = 0;
         else
            ++*rxo_index_ptr;
         if (*txi_index_ptr >= ubuf_sz)
            *txi_index_ptr = 0;
         else
            ++*txi_index_ptr;
      }

      /* Wait for any previous transmit request to run down, just for fun */
      while (!txflag);                 /* Wait for transmit ring empty */
      txflag = 0;

      /* Request transmission */
      GSTLO |= REGBSY;                 /* Preset registers busy */
      TX |= (1 << channel);            /* Turn on the transmitter */
                                       /* If using more than 1 channel,
                                       modify TX register using indivisible
                                       RMW instructions*/
      while (GSTLO & REGBSY);          /* Wait for registers not busy */
   }
}

void
setvec(Long vecno, void (*israddr)(void))
{
   asm(" move.l    8(a6),d0           get interrupt vector#  ");
   asm(" move.l    12(a6),a0          get pointer to handler  ");
   asm(" dc.w      $a116              vmeprom call: xvec     ");
}

#pragma interrupt()

void
err_isr()
{
   printf("Error interrupt, GST = %04X\n", GST);
   abort();
}
#pragma interrupt()

void
chn_isr()
{
   if (CST & URCV_RDY) {
```

```
      rxflag = 1;
      CLRBITW (CST, URCV_RDY);      /* Use indivisible RMW instruction to
                                    CLR URCV_RDY*/
   }
   if (CST & UTX_RDY) {
      txflag = 1;
      CLRBITW (CST, UTX_RDY);       /* Use indivisible RMW instruction to
                                    CLR UTX_RDY*/
   }
}
```

4.6 RUNNING SELF-TESTS FROM THE HOST

The VMIVME-6016 can perform several diagnostic tests on parts of its hardware. The host CPU must set up each test and command it to run. The CPU uses the Self-Test (SLFTST) bit in the CR0 register, and the STPROC register to control the self-tests. The first two steps in running a self-test are the same for each test; they are shown as *a* and *b* below. From that point, the tests differ. After starting a test, the host CPU should wait for either SLFTST or STBUSY to go to zero as the indication that the test has finished. After a test has completed, the STFAIL bit in GST indicates success or failure. Some of the tests may return a code in CST[0] (the CST register for channel zero) on failure.

```
a. Make sure the GO and BREAK registers are all zeros.
b. Set the desired test in the STPROC register.
TEST_DMA_R: VMEbus DMA read (bus --> board) test
              Load source pointer in BUFBASE
              Set SZ_AM
              Set byte count in BREAK (multiple of 4)
              Set SLFTST
                  (BREAK, if nonzero, is the residual byte count)
TEST_DMA_W: VMEbus DMA write (board --> bus) test
              Load destination pointer in BUFBASE
              Set SZ_AM
              Set byte count in BREAK (multiple of 4)
              Set SLFTST
                  (BREAK, if nonzero, is the residual byte count)
TEST_BERR:  Self-access BERR test
              Set base address of board in BUFBASE
              Set SZ_AM
              Set SLFTST
                  (BUFBASE contains longword read if did not BERR)
TEST_UART:  UART port test (local loopback)
              Set SLFTST
TEST_TIMER: Timer interrupt test
              Set SLFTST
                  (Wait about 5 milliseconds before testing SLFTST)
TEST_VINT:  VMEbus interrupt test
              Set vector in ER_VEC
              Set level (1-7) in ER_MSK[ERLEVEL], other bits zeros
              Set SLFTST (causes the VMEbus interrupt)
                  (VMIVME-6016 waits at most 10 milliseconds)
```

Table 4.6-1. CST[0] Register Self-Test Failure Codes

Mnemonic	Value	Reason for Failure
DMA_GET_ERR	1	TEST_DMA data read error
DMA_PUT_ERR	2	TEST_DMA data write error
BERR_NO_BERR	3	TEST_BERR did not BERR
TIMER_FAST	4	TEST_TIMER: timer speed high
TIMER_SLOW	5	TEST_TIMER: timer speed low
VINT_TIMEOUT	6	TEST_VINT timed out
VINT_ILLEGAL	7	TEST_VINT improperly set up
BERR_TIMEOUT	8	TEST_BERR timed out without self-access
UART_BAD	1X	TEST_UART channel X bad
TEST_UNDEFINED	255	Undefined test bit in STPROC

M6016/T4.6-1

Table 4.6-2. Test Bits in STPROC Register

Code	Bit	Description
TEST_DMA_R	Bit 5	VMEbus DMA read (bus --> board) test
TEST_DMA_W	Bit 4	VMEbus DMA write (board --> bus) test
TEST_BERR	Bit 3	Self-access BERR test
TEST_UART	Bit 2	UART test (local loopback)
TEST_TIMER	Bit 1	Timer test
TEST_VINT	Bit 0	VMEbus interrupt test

M6016/T4.6-2

4.7 SAMPLE HEADER FILE

```
/* File:           6016hdwe.h
 * Description:    VMIVME-6016 register definitions
 * Revision:       XO
 * Date:           21 Jul 94
 * Where used:     Force-33 CPU test code for, and firmware for VMIVME-6016
 * Property of:    VMIC (VME Microsystems International Inc.)
 */

/*
 * The Global "registers" as based lvalues.
 * These take up thirty-two bytes, at the beginning of the
 * 256 bytes of A16 (short I/O) memory.
 */

#define GREG_SIZE       0x20            /* Room for global registers */
```

```
#define BRD_ID  *(Byte *)(regbase+0x00) /* Board ID */
#define STFLAG  *(Byte *)(regbase+0x01) /* Self test flag in ID register */
#define OPTIONS *(Word *)(regbase+0x02) /* ROM version code and hdwe options
*/
#define CR0     *(Byte *)(regbase+0x04) /* Command register 0 */
#define CR1     *(Byte *)(regbase+0x05) /* Command register 1 */
#define GO     *(Word *)(regbase+0x06) /* "GO" bits, ch[15..0] = GO[15..0] */
#define TX     *(Word *)(regbase+0x08) /* "TX" bits, ch[15..0] = TX[15..0] */
#define RX     *(Word *)(regbase+0x0A) /* "RX" bits, ch[15..0] = RX[15..0] */
#define BREAK   *(Word *)(regbase+0x0C) /* "BREAK", ch[15..0] = BREAK[15..0]
*/
#define CR2     *(Byte *)(regbase+0x0E) /* Command register 2 */
                                         /* 0F reserved, must be zero */
#define SZ_AM   *(Byte *)(regbase+0x10) /* Master's size and AM */
#define ST_PROC *(Byte *)(regbase+0x11) /* Selftest procedure register */
#define ER_MSK  *(Byte *)(regbase+0x12) /* Error interrupt control */
#define ER_VEC  *(Byte *)(regbase+0x13) /* Error interrupt vector */
#define BUFBASE *(Long *)(regbase+0x14) /* Base of buffers (M) or board (S)
*/
#define GST     *(Word *)(regbase+0x18) /* Global Status register */
#define MAS_GRN *(Byte *)(regbase+0x1A) /* Master Granularity */
                                       /* 1B .. 1F reserved, must be zero */

#define GSTLO *(Byte*)(regbase+0x19) /* Lower byte of GST Register*/

/* CR0 bits */
#define MASTER  0x80    /* 1=Master (buffers off-board), 0=Slave mode */
#define SLV_OK  0x40    /* 1=A24/A32 slave decode OK */
#define SLV_32  0x20    /* 1=slave is A32, 0=slave is A24 */
#define SUPONLY 0x10    /* 1=slave is SUP only, 0=slave is SUP and NP */
#define TXBRK   0x08    /* 1=Send BREAK signals, as marked in BREAK */
#define SLFTST  0x04    /* 1=selftest, 0=normal */
#define RST_BRD 0x02    /* 1=reset entire board, 0=normal (run) */
#define LEDN    0x01    /* 1=LED off, 0=LED on */

/* CR1 bits */
#define ARBMOD  0x08    /* SCON VMEbus Arbiter mode: 1=Priority 0=RR */
#define BERR_TO 0x07    /* VMEbus BERR timeout if SCON */

/* CR2 bits */
#define BRLEVEL 0xC0    /* VMEbus Bus Request Level to use */
#define RELMOD  0x30    /* VMEbus Release Mode */
#define RELROR  0x00            /* Release on Request */
#define RELRWD  0x10            /* Release when Done */
#define RELROC  0x20            /* Release on BCLR* */
#define RELBCAP 0x30            /* Bus capture and hold */
#define FAIR_TO 0x0F    /* Fairness control and timeout */

/* SZ_AM: Master's size and AM bits */
#define VMESIZE         0xC0  /* VMEbus size, 00 = L, 01 = B, 10 = W */
#define VME_SIZE_32     0x00    /* 32-bit size as VMEbus master */
#define VME_SIZE_16     0x80    /* 16-bit size as VMEbus master */
#define VME_SIZE_8      0x40    /*  8-bit size as VMEbus master */
#define ADDRMOD         0x3F  /* VMEbus Address Modifier */
```

```
/* ST_PROC: Selftest procedure bits */
#define TEST_DMA_R      0x20    /* VMEbus DMA read (bus --> board) test */
#define TEST_DMA_W      0x10    /* VMEbus DMA write (board --> bus) test */
#define TEST_BERR       0x08    /* Self-access BERR test */
#define TEST_UART       0x04    /* UART test (local loopback) */
#define TEST_TIMER      0x02    /* Timer test */
#define TEST_VINT       0x01    /* VMEbus interrupt test */

/* ER_MSK: Error interrupt control register */
#define REGBUSY_MSK     0x80    /* Write-to-register handled mask */
#define RFNZ_MSK        0x40    /* Reserved-field-non-zero mask */
#define ST_FAIL_MSK     0x20    /* Selftest error mask */
#define HDWE_ERR_MSK    0x10    /* Hardware Error mask */
#define BERR_MSK        0x08    /* VMEbus BERR mask */
#define ER_LEVEL        0x07    /* VMEbus level code */

/* GST (Board Status) bits */
#define ANYERR  0x8000  /* Any Error (OR of bits marked ERROR) */
#define GLBILL  0x4000  /* Illegal content in a global register */
#define STBUSY  0x2000  /* Self test function busy */
#define SCON    0x1000  /* 1 = SCON jumper is installed (slot 1) */
#define RFNZ    0x0800  /* Reserved Field set non-zero (ERROR) */
#define STFAIL  0x0400  /* 6016 selftest error (ERROR) */
#define HDWEERR 0x0200  /* 6016 hardware error (ERROR) */
#define BERR    0x0100  /* BERR on VMEbus master cycle (ERROR) */
#define REGBUSY 0x0080  /* Write-to-register process busy */
#define REGBSY  0x80    /* BUSY MASK for low byte of GST Register*/

/*
 * The Channel Control Block "registers" as offsets from an A register.
 * There are sixteen sets of these, each using fourteen bytes.  They
 * immediately follow the Global "registers", exactly using up
 * all 256 bytes of A16 (short I/O) memory.
 */

#define CREG_SIZE       0x0E            /* Room for one CCB */
#define CST     *(Word *)(chnbase+0x00) /* Channel Status */
#define CH_MSK  *(Byte *)(chnbase+0x02) /* Channel Interrupt Mask */
#define CH_VEC  *(Byte *)(chnbase+0x03) /* Channel Interrupt Vector */
#define EOB     *(Byte *)(chnbase+0x04) /* End-of-block code */
#define XOFF    *(Byte *)(chnbase+0x05) /* Flow control XOFF code */
#define XON     *(Byte *)(chnbase+0x06) /* Flow control XON code */
#define BRK_DUR *(Byte *)(chnbase+0x07) /* BREAK duration in ms +0/-1 */
#define SZ_RING *(Byte *)(chnbase+0x08) /* Internal Ring size */
#define LO_RING *(Byte *)(chnbase+0x09) /* Internal Ring low water mark */
#define HI_RING *(Byte *)(chnbase+0x0A) /* Internal Ring high water mark */
#define CH_CON1 *(Byte *)(chnbase+0x0B) /* Channel Control Byte 1 */
#define CH_CON2 *(Byte *)(chnbase+0x0C) /* Channel Control Byte 2 */
#define SZ_UBUF *(Byte *)(chnbase+0x0D) /* User Buffer Size */

/* CST bits */
#define DCD_COS     0x8000      /* DCD (Data-Carrier-Detect) changed state */
#define CTS_COS     0x4000      /* CTS (Clear-To-Send) changed state */
#define DCD         0x2000      /* DCD state */
```

```
#define CTS         0x1000    /* CTS state */
#define INTBSY      0x0800    /* Host's interrupt routine busy */
#define ILL         0x0400    /* Illegal BAUD or PAR_TYP, or other param */
#define URCV_RDY    0x0200    /* User receive buffer has data */
#define UTX_RDY     0x0100    /* User transmit buffer can take data */
#define BRK_RCVD    0x0080    /* BREAK received */
#define FE          0x0040    /* Framing Error */
#define PE          0x0020    /* Parity Error */
#define OVERRUN     0x0010    /* Receiver Overrun */
#define TXEMT       0x0008    /* Transmitter Empty */
#define TXRDY       0x0004    /* Transmitter Ready (but maybe not empty) */
#define FFUL        0x0002    /* Receiver FIFO full */
#define RXRDY       0x0001    /* Receiver Ready (has a character) */

/* CH_MSK bits */
#define COS_MSK     0x80      /* CTS or DCD changed state */
#define ERR_MSK     0x40      /* FE, PE, OVERRUN, or illegal code */
#define BREAK_MSK   0x20      /* BREAK received */
#define RCV_MSK     0x10      /* User receive buffer has data */
#define TX_MSK      0x08      /* User transmit buffer needs data */
#define CH_LEVEL    0x07      /* Channel Interrupt Level */

/* CH_CON1 bits */
#define UB_RING          0x80   /* User Buffer type: 1=ring 0=linear */
#define FLOWCON          0x60   /* Flow control:                     */
                                /*      00: None                     */
                                /*      01: XON/XOFF                 */
                                /*      02: Any/XOFF                 */
                                /*      03: RTS/CTS and DCD          */
#define BAUD             0x1F   /* Baud rate for both TX and RX:          */
                                /*      00000: 50       10000: 75         */
                                /*      00001: 110      10001: 110        */
                                /*      00010: 134.5    10010: 38.4K      */
                                /*      00011: 200      10011: 150        */
                                /*      00100: 300      10100: 300        */
                                /*      00101: 600      10101: 600        */
                                /*      00110: 1200     10110: 1200       */
                                /*      00111: 1050     10111: 2000       */
                                /*      01000: 2400     11000: 2400       */
                                /*      01001: 4800     11001: 4800       */
                                /*      01010: 7200     11010: 1800       */
                                /*      01011: 9600     11011: 9600       */
                                /*      01100: 38.4K    11100: 19.2K      */

/* CH_CON2 bits */
#define INP_TIMEOUT      0xC0   /* Input timeout control                 */
#define INP_T_NONE       0x00   /*      00 : none                        */
#define INP_T_3CH        0x40   /*      01 : 3 char times or break       */
                                /*           duration, whichever is >    */
#define INP_T_HALFSEC    0x80   /*      10: 1/2 second                   */
#define INP_T_SEC        0xC0   /*      11: 1 second                     */
#define TWO_STOP         0x20   /* Transmit stop bits: NZ=2, Z=1         */
#define PAR_TYP          0x18   /* Transmit/Receive Parity Type:         */
                                /*      00: Odd or Even, use PAR_ODD     */
```

```
                                /*      01: Force to PAR_ODD state     */
                                /*      10: No parity bit              */
#define PAR_ODD         0x04    /* Parity: 1=odd 0=even                */
#define SZ_CHAR         0x03    /* Bits/char not including parity bit: */
                                /*      00: 5                          */
                                /*      01: 6                          */
                                /*      10: 7                          */
                                /*      11: 8                          */

/* End of 6016hdwe.h */
```

4.7.1 Sample Header File for Indivisible RMWs

```
/* File:          setclr.h
 * Description:   SET and CLR macros using CAS instructions
 * Revision:      XO
 * Date:          06 Sep 95
 * Where used:    VMIVME-6016 CrossCode-C Test Code running on Force 33
 * Property of:   VMIC (VME Microsystems International Corporation)
 *
 *
 * Revision history:
 *   06 Sep 95  Initial
 */

/* Set bits in 16-bit shared memory reg using RMC */
#define SETBITW(reg, bits)      set_bw(&(reg), bits)

/* Set bits in 8-bit shared memory reg using RMC */
#define SETBITB(reg, bits)      set_bb(&(reg), bits)

/* Clear bits in 16-bit shared memory reg using RMC */
#define CLRBITW(reg, bits)      clr_bw(&(reg), bits)

/* Clear bits in 8-bit shared memory reg using RMC */
#define CLRBITB(reg, bits)      clr_bb(&(reg), bits)

void set_bw(Word * reg, int bits);
void set_bb(Byte * reg, int bits);
void clr_bw(Word * reg, int bits);
void clr_bb(Byte * reg, int bits);
```

4.7.2 Assembler Source File for 680x0 RMW Instructions

```
; File:           setclr.s
; Description:    SET and CLR routines using CAS instructions
; Revision:       XO
; Date:           07 Sep 95
; Where used:     VMIVME-6016 CrossCode-C Test Code running on Force 33
; Property of:    VMIC (VME Microsystems International Corporation)
;
;
; Revision history:
;   07 Sep 95  Initial
;
; Set bits in 16-bit shared memory reg using RMC
; void set_bw(Word * reg, int bits);
;
        .OPTION target=68020/68881,flags=gG
;
        xdef    _set_bw
        xdef    _set_bb
        xdef    _clr_bw
        xdef    _clr_bb
;
        section code
;
_set_bw:                        ; SETBITW
        move.l  d2,-(sp)        ; Save a register
        move.l  8(sp),a0        ; &reg -> A0
        move.l  12(sp),d2       ; Bits to set -> D2
        move.w  (a0),d0         ; Old reg contents -> D0
        move.w  d2,d1           ; CAS loop: Copy bits -> D1
        or.w    d0,d1           ; OR in the old contents
        cas.w   d0,d1, (a0)     ; Try to set bits in reg
        bne.s   *-8             ; Keep trying until it works
        move.l  (sp)+,d2        ; Restore the saved register
        rts                     ; Return to caller
;
_set_bb:                        ; SETBITW
        move.l  d2,-(sp)        ; Save a register
        move.l  8(sp),a0        ; &reg -> A0
        move.l  12(sp),d2       ; Bits to set -> D2
        move.b  (a0),d0         ; Old reg contents -> D0
        move.b  d2,d1           ; CAS loop: Copy bits -> D1
        or.w    d0,d1           ; OR in the old contents
        cas.b   d0,d1, (a0)     ; Try to set bits in reg
        bne.s   *-8             ; Keep trying until it works
        move.l  (sp)+,d2        ; Restore the saved register
        rts                     ; Return to caller
;
_clr_bw:                        ; CLRBITW
        move.l  d2,-(sp)        ; Save a register
        move.l  8(sp),a0        ; &reg -> A0
        move.l  12(sp),d2       ; Bits to clear -> D2
        not.w   d2              ; Invert bits for mask
        move.w  (a0),d0         ; Old reg contents -> D0
        move.w  d2,d1           ; CAS loop: Copy bits -> D1
        and.w   d0,d1           ; Mask in the old contents
        cas.w   d0,d1, (a0)     ; Try to set bits in reg
```

```
        bne.s   *-8             ; Keep trying until it works
        move.l  (sp)+,d2        ; Restore the saved register
        rts                     ; Return to caller
;
_clr_bb:                        ; CLRBITB
        move.l  d2,-(sp)        ; Save a register
        move.l  8(sp),a0        ; &reg -> A0
        move.l  12(sp),d2       ; Bits to clear -> D2
        not.b   d2              ; Invert bits for mask
        move.b  (a0),d0         ; Old reg contents -> D0
        move.b  d2,d1           ; CAS loop: Copy bits -> D1
        and.b   d0,d1           ; Mask in the old contents
        cas.b   d0,d1, (a0)     ; Try to set bits in reg
        bne.s   *-8             ; Keep trying until it works
        move.l  (sp)+,d2        ; Restore the saved register
        rts                     ; Return to caller
;
```

SECTION 5

CONFIGURATION AND INSTALLATION

5.1 UNPACKING PROCEDURES

CAUTION

SOME OF THE COMPONENTS ASSEMBLED ON VMIC'S PRODUCTS MAY BE SENSITIVE TO ELECTROSTATIC DISCHARGE AND DAMAGE MAY OCCUR ON BOARDS THAT ARE SUBJECTED TO A HIGH ENERGY ELECTROSTATIC FIELD. WHEN THE BOARD IS PLACED ON A BENCH FOR CONFIGURING, ETC., IT IS SUGGESTED THAT CONDUCTIVE MATERIAL SHOULD BE INSERTED UNDER THE BOARD TO PROVIDE A CONDUCTIVE SHUNT. UNUSED BOARDS SHOULD BE STORED IN THE SAME PROTECTIVE BOXES IN WHICH THEY WERE SHIPPED.

Upon receipt, any precautions found in the shipping container should be observed. All items should be carefully unpacked and thoroughly inspected for damage that might have occurred during shipment. The board(s) should be checked for broken components, damaged printed circuit board(s), heat damage, and other visible contamination. All claims arising from shipping damage should be filed with the carrier and a complete report sent to VMIC together with a request for advice concerning the disposition of the damaged item(s).

5.2 PHYSICAL INSTALLATION

CAUTION

DO NOT INSTALL OR REMOVE THE BOARD WHILE THE POWER IS APPLIED.

De-energize the equipment and insert the board into an appropriate slot of the chassis. While ensuring that the board is properly aligned and oriented in the supporting card guides, slide the board smoothly forward against the mating connector until firmly seated.

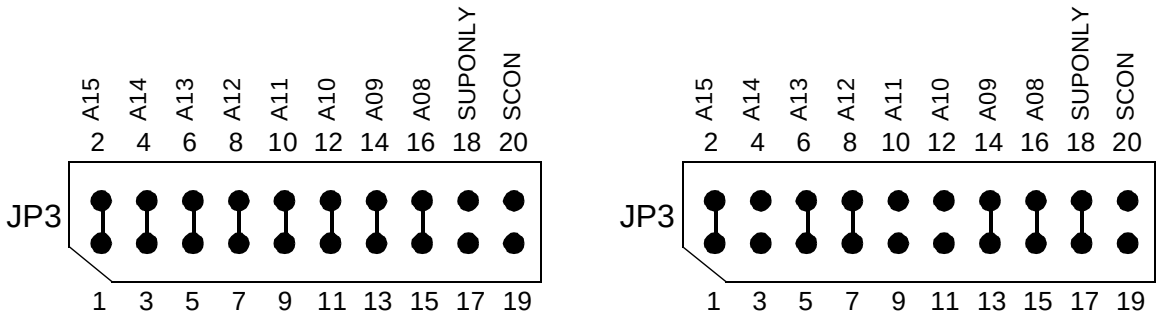
5.3 JUMPER CONFIGURATIONS

The VMIVME-6016 has several jumper fields that are preconfigured for certain defaults at the factory but may need to be changed according to the host system configuration. Figure 5.3-1 on page 5-4 shows the locations of all jumpers on the VMIVME-6016. Figure 5.3-2 on page 5-5 shows the front panel. Figure 5.3-3 and Table 5.3-1 on page 5-5 show the pinout for the RJ12 sockets.

5.3.1 VMEbus Address Configuration

The VMIVME-6016 registers occupy 256 bytes of memory and must be addressed in short I/O (A16) mode. Jumpers A15 through A08 at JP3 determine the VMEbus address of the VMIVME-6016 registers and Control Blocks. An installed jumper equals zero (0); an omitted jumper equals one (1).

The factory default configuration has the SUPONLY jumper (pins 17 and 18 of JP3) removed, allowing the board to respond to both supervisory and nonprivileged accesses. If the SUPONLY jumper is installed, the board responds to only supervisory accesses.



Factory Configuration

Base Address $0000
Both Supervisory and
Nonprivileged Accesses Allowed

Example Configuration

Base Address $4C00
Supervisory Accesses Only

*ON = 1; OFF = 0

M6016/F5.3.1-1

Figure 5.3.1-1. Base Address and Access Mode Selection

5.3.2 System Controller Configuration

If the VMIVME-6016 is installed as the system controller in slot 1 of the chassis, the board must have SCON jumper (pins 19 and 20 of JP3) installed. If the SCON jumper is installed, the VMIVME-6016 performs the following system controller functions:

a. Full-function, four-level bus arbiter. The arbitration method used can be toggled between a round-robin and a prioritized scheme by changing the Arbiter Mode bit in the CR1 register (see Section 4.2.5).

b. The 16 MHz SYSCLK VMEbus system clock.

c. The SYSRESET* system reset signal.

d. IACK* interrupt acknowledge daisy-chain driver.

e. VMEbus BERR bus error timer. By default, the VMIVME-6016 activates the BERR* signal when any bus master fails to get a DTACK* acknowledge signal within 64 microseconds. This bus error time-out can be changed. See Section 4.2.5.

5.3.3 Hardware Reset

JP1, if momentarily shorted, causes a hard reset just as if power had been removed and restored. JP1 should normally be left unconnected.

5.3.4 Fixed Jumpers

JP2 is used only during special factory test procedures. JP2 should normally be left unconnected. JP4 is configured at the factory and must not be changed.

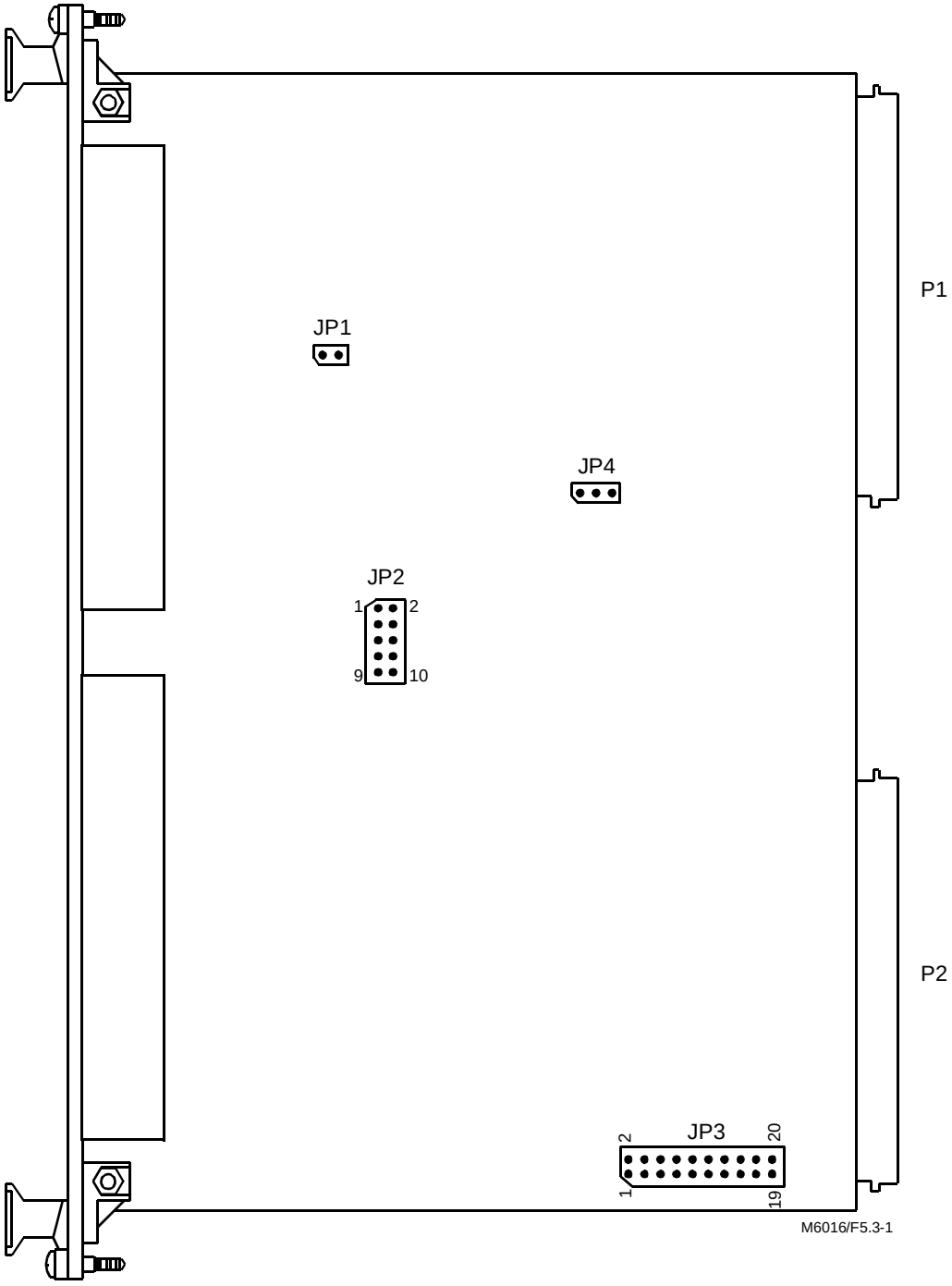
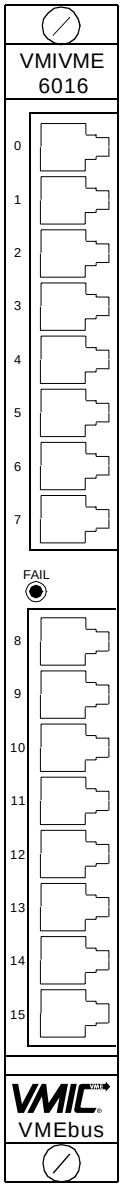
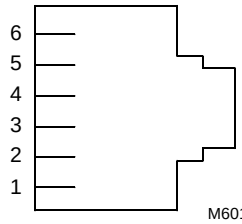


Figure 5.3-1. VMIVME-6016 Jumper Field Locations



M6016/F5.3-2

Figure 5.3-2. VMIVME-6016 Front Panel



M6016/F5.3-3

Figure 5.3-3. RJ12 Socket

Table 5.3-1. RJ12 Socket Pinout

Signal	RJ12	Meaning	Direction
CTS	1	Clear to Send	Input
GND	2	Signal Ground	
RXD	3	Receive Data	Input
TXD	4	Transmit Data	Output
DCD	5	Data Carrier Detect	Input
RTS	6	Request to Send	Output

M6016/T5.3-1

SECTION 6

MAINTENANCE

6.1 MAINTENANCE

This section provides information relative to the care and maintenance of VMIC's products. If the products malfunction, verify the following:

a. Software
b. System configuration
c. Electrical connections
d. Jumper or configuration options
e. Boards are fully inserted into their proper connector location
f. Connector pins are clean and free from contamination
g. No components of adjacent boards are disturbed when inserting or removing the board from the chassis
h. Quality of cables and I/O connections

If products must be returned, contact VMIC for a Return Material Authorization (RMA) Number. **This RMA Number must be obtained prior to any return**.

6.2 MAINTENANCE PRINTS

User level repairs are not recommended. The appendix to this manual contains drawings and diagrams for reference purposes only.