XVME-566

FAST ANALOG INPUT MODULE

P/N 74566-002B

© 1993 XYCOM, INC.

Printed in the United States of America

XYCOM
750 North Maple Road
Saline, Michigan 48176
(313) 429-4971

Xycom Revision Record

Revision	*Description*	*Date*
A	Manual Released	5/87
B	Incorporated PCNs 117 and 143	6/93

Copyright Information
This document is copyrighted by Xycom Incorporated (Xycom) and shall not be reproduced or copied without expressed written authorization from Xycom.

The information contained within this document is subject to change without notice. Xycom does not guarantee the accuracy of the information and makes no commitment to keeping it up to date.

Address comments concerning this manual to:

xycom
Technical Publications Department
750 North Maple Road
Saline, Michigan 48176

Part Number: ***74566-002B***

TABLE OF CONTENTS

Chapter 1

MODULE DESCRIPTION

1.1 INTRODUCTION

The XVME-566 Fast Analog Input Module is a high-performance, VMEbus[a]compatible, 6U (double-high) module capable of performing analog-to-digital conversions with 12-bit resolution and 12-bit accuracy. This module provides 32 single-ended (SE) or 16 differential input (DI) analog input channels.

All analog input channels on the module can be configured for unipolar and bipolar voltage (0-10V, ±5V and ±10V). Each input channel is programmable to a specific gain within one of three jumper-selected gain ranges (1 to 10, 4 to 40, and 10 to 100). Gains within the ranges (1, 2, 5 and 10) are stored within a 32-byte Gain RAM.

Collection of up to 32K 12-bit samples (32767) are possible via the dual-ported, 64Kbyte Data RAM. The module's dual sample-and-hold architecture assures 12-bit data conversion times of 10usec and a sample throughput rate of 100KHz (8-bit resolution conversion times of 7usec allows 144KHz throughput).

The module can operate in one of two data acquisition modes, Sequential or Random. In the Sequential operating mode, data is sampled, converted and stored in the sequence specified in the Sequence RAM. In the Random operating mode, data acquisition is performed on one channel at a time with a throughput of 50 KHz.

Either of two data storage modes (contiguous or image) can be used. The contiguous mode stores data from a specified start address, permitting different values to be accumulated for each input channel, therefore creating a history of the events on each channel. The image mode, in contrast, provides only the latest (or updated) value on a channel.

The 256-byte Sequence RAM allows the user to specify the order in which data acquisition is performed on the input channels. Once initialization has occurred, the XVME-566 is capable of gathering data without operator intervention.

The Am9513A System Timing Controller (STC) on the module can generate outputs that can cause interrupts to the VMEbus, allowing added flexibility during data acquisition. Those features of the STC utilized most often by the XVME-566 are the Sample Clock, Trigger Clock and Event Counter. Detailed information for the System Timing Controller is available with the AM9513 Handbook included with the module.

1.2 MANUAL STRUCTURE

The first chapter is an overview introducing the user to the XVME-566 general specifications and functional capabilities. Successive chapters develop the various aspects of module specifications and operation in the following manner:

Chapter One - Module Description: a general discussion of the module including complete functional and environmental specifications, VMEbus compliance information and detailed block diagrams

Chapter Two - Installation: module configuration information covering specific system requirements, jumpers, switches and connector pinouts

Chapter Three - Programming Overview: a brief description of the programming environment for the module

Chapter Four - Programming: information required to program the module for analog input operation

Chapter Five - Calibration: procedures for analog input calibration

Appendix A - Xycom Standard I/O Architecture: background information describing the standard I/O hardware that is relevant to the XVME-566

Appendix B - VMEbus Connector/Pin Description: listings and descriptions of the VMEbus signals, connectors, pin numbers and their descriptions

Appendix C - Quick Reference Guide: (blue pages) compact reference to tables and figures containing preliminary information (jumpers, module addresses, memory map, etc)

Appendix D - Diagrams and Schematics: module block diagrams and schematics

1.3 RELATED DOCUMENTS

The following document should prove helpful in the understanding of the operation of the XVME-566 Fast Analog Input Module:

Publisher	Title	Date
Advanced Micro Devices	Handbook, The Am9513 System Timing Controller*	1986

*Xycom part no. 91366-001 (one copy provided with each XVME-566 board)

1.4 MODULE OPERATIONAL DESCRIPTION

Figure 1-1 shows an operational block diagram of the XVME-566 Fast Analog Input Module.

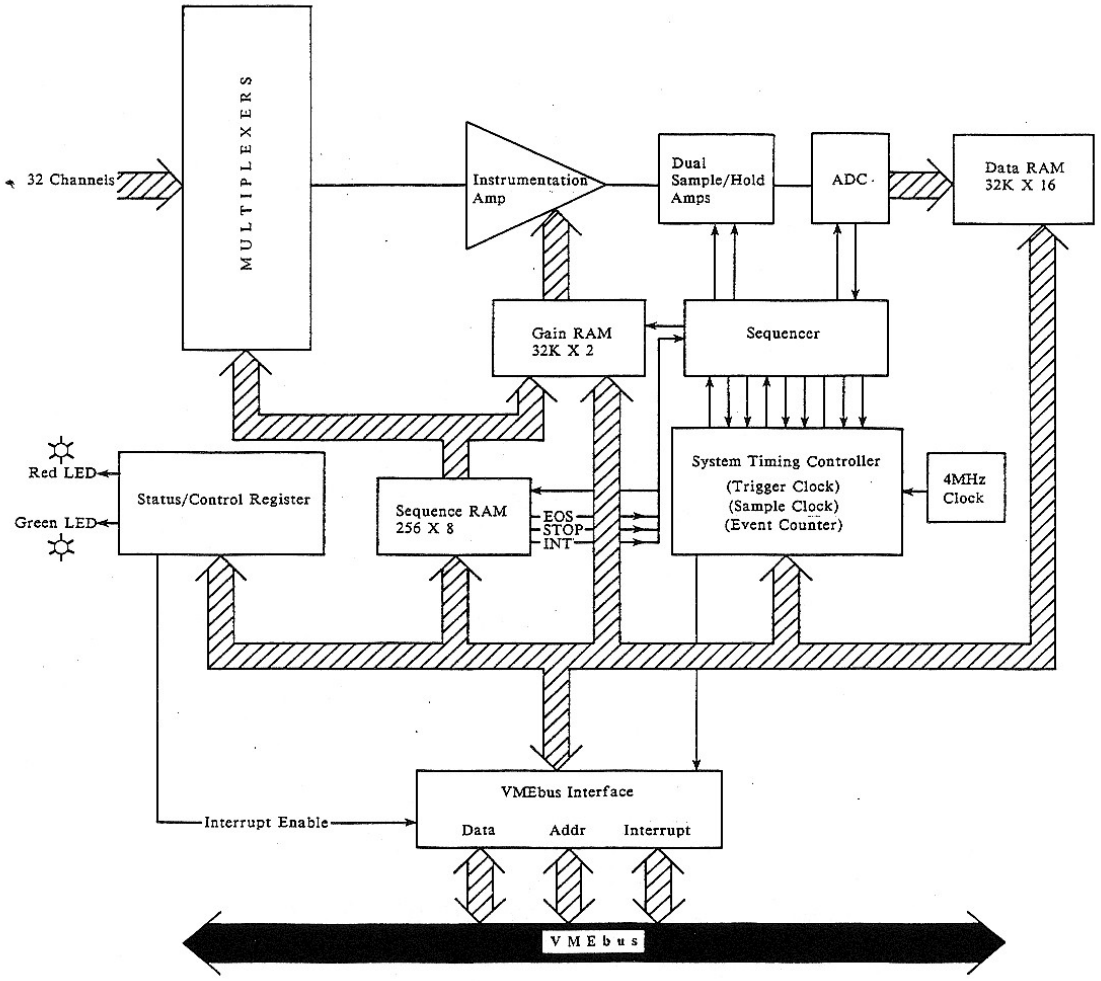


Figure 1-1

XVME-566 Module Block Diagram

1.4.1 Xycom Non-Intelligent Kernel

The Non-intelligent Kernel provides an interface between application circuitry and the VMEbus. Among its provisions are the necessary circuitry to receive and generate signals (required by the VMEbus specification, Revision C.1) for a 16-bit slave. The addition of the analog-to-digital circuitry makes the XVME module complete.

The XVME-566 Non-intelligent Kernel has been modified to accommodate 64K bytes of Data RAM in the standard access mode. This does not alter the I/O functions access (standard or Short I/O) provided by Xycom's module architecture.

The following functional circuits are included in the kernel design:

-- Control and Address Buffers
-- Address Decode Circuitry
-- Status/Control Register Port (VMEbus accessible, 16 bits)
-- Module Identification Information (64 bytes)
-- Full VMEbus slave
-- Interrupter (any of I (1-7), static)

The Xycom Non-Intelligent Kernel is described in further detail in Appendix A.

1.4.2 Features of Xycom's Standard I/O Architecture

The XVME-566 conforms to the Xycom Standard I/O Architecture. In addition, the XVME-566 may be accessed from the VMEbus short I/O or standard address space (except the Data RAM, which always resides in the standard address space). The following features apply to the operation of this module:

Module Address:	The module can be located at any one of 64 base addresses in VMEbus Short I/O address space.
Module Address Space:	The module occupies 1K of Short I/O Address Space known as the I/O Interface Block. All of its programming registers are located within this block.
Module I.D.	The module has I.D. information which provides its name model number, manufacturer and revision level at a location that is consistent with other Xycom modules.
Status/Control Register	This register is always located at module base address +81H. The lower four bits (interrupt pending, interrupt enable, red and green LED bits) are standard from module to module.

A detailed description of Xycom I/O Architecture is presented in Appendix A of this manual.

1.4.3 Application Circuitry

As Figure 1-1 shows, the analog-to-digital circuitry on the XVME-566 consists of the following parts:

-- VMEbus interface circuitry
-- Gain RAM
-- Sequence RAM
-- Data RAM
-- Multiplexers
-- Analog-to-digital converters
-- Dual sample/hold architecture
-- Programmable gain
-- Channel mode & control logic
-- Timing Sequencer

1.5 SPECIFICATIONS

1.5.1 Mechanical Specifications

The XVME-566 module uses the standard Xycom front panel. The handles are mounted at the top and bottom of the front panel. The analog inputs are connected in the front of the panel (via one 50-pin flat-ribbon connector) to JK1. Figure 1-2 illustrates the front panel of the module.

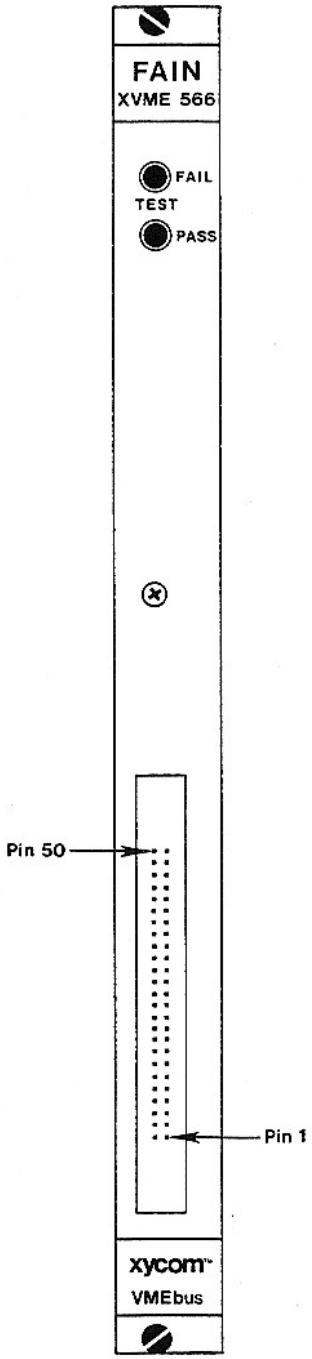


Figure 1-2

XVME-566 Front Panel

Table 1-1

XVME-566 Module Specifications

1.5.2 Electrical Specifications

Characteristic	Specification
Number of Channels	
Single-Ended (SE)	32
Differential (DI)	16
Board Requirements	
Supply Voltage	±5VDC, ±5%
Supply Current	2.1 Amp
Analog Inputs	
Analog-to-Digital Input	
Full-scale Voltage Ranges	
Unipolar	0-10V
Bipolar	±5V, ±10V
Programmable Gain	
Range 1	1,2,5, or 10
Range 2	4,8,20,or 40
Range 3	10,20,50, or 100
Maximum Input Voltage	
Power On	±35V
Power Off	±20V
Input Impedance	10Mohm min.
Bias Current	±100nA max
Input Capacitance	100pf max
Operating Common Mode Voltage	14V
Accuracy	
Resolution	12 bits
Linearity	±1/2 LSB
Differential Linearity	±1/2 LSB
Monotonicity	Guaranteed
System Accuracy	
Gain = 1	±0.01% FSR max
Gain = 10	±0.1% FSR max
Gain = 100	±0.1% FSR max

1.5.2 Electrical Specifications (cont)

Characteristic	Specification	
System Accuracy Temperature Drift		
Gain = 1	40 ppm/Degree C max	
Gain = 10	75 ppm/Degree C max	
Gain = 100	110 ppm/Degree C max	
Common Mode Rejection Ratio	60db min	
Speed	12-bit	8-bit
Conversion Time	10uSec	7usec
Throughput Frequency	≤ 100KHz	≤ 144KHz

1.5.2.1 Mating Connector Information

Characteristic	Specification
50-Pin Ribbon Header	
3M Part No. 3425-6000	without strain relief
3M Part No. 3425-6050	with strain relief
Amp Part No. 1-102387-0	discrete connector
Amp Part No. 87667-4	crimp for discrete connector

1.5.3 Environmental Specifications

Characteristic		Specification
Temperature		
	Operating	0 to 65 C
	Non-operating	-40 to 85 C
Humidity		
	Operating	5 to 95% RH, non-condensing
Shock		
	Operating	30g peak acceleration 11mSec duration
	Non-operating	50g peak acceleration 11mSec duration
Vibration		
	Operating	5 to 2000Hz .015 in. peak-to-peak 2.5g max
	Non-operating	5 to 2000Hz .030 in. peak-to-peak 5.0g max

1.5.4 VMEbus Compliance

* Complies with VMEbus specification Revision C.1
* A16:D16 DTB Slave
* Interrupter - I(1-7) [static]
* Interrupter Vector - Programmable
* AM codes 29,2D,39,3D, response [static]
* NEXP board size (160mm x 233.35mm)
* Conforms to Xycom Standard I/O Architecture

Chapter 2

IDENTIFICATION AND CONFIGURATION

2.1 IDENTIFICATION AND CONFIGURATION

2.1 GENERAL

This chapter describes the various components on the XVME-566, and includes necessary information for configuring the module before start-up.

2.2 SYSTEM REQUIREMENTS

The Fast Analog Input Module is a double-high VMEbus-compatible module. To operate, it must be properly installed in a VMEbus backplane. The minimum system requirements for operation of the module are one of the following (A or B):

A) A host processor installed in the same backplane

AND

A properly installed controller subsystem. An example of such a subsystem is the XYCOM XVME-010 System Resource Module.

OR

B) A host processor which incorporates an on-board controller subsystem (such as the XVME-600 or XVME-601 68000 Processor Module).

2.3 LOCATIONS OF COMPONENTS RELEVANT TO INSTALLATION

The jumpers, switches, test points, calibration potentiometers and connectors on the module are illustrated in Figure 2-1.

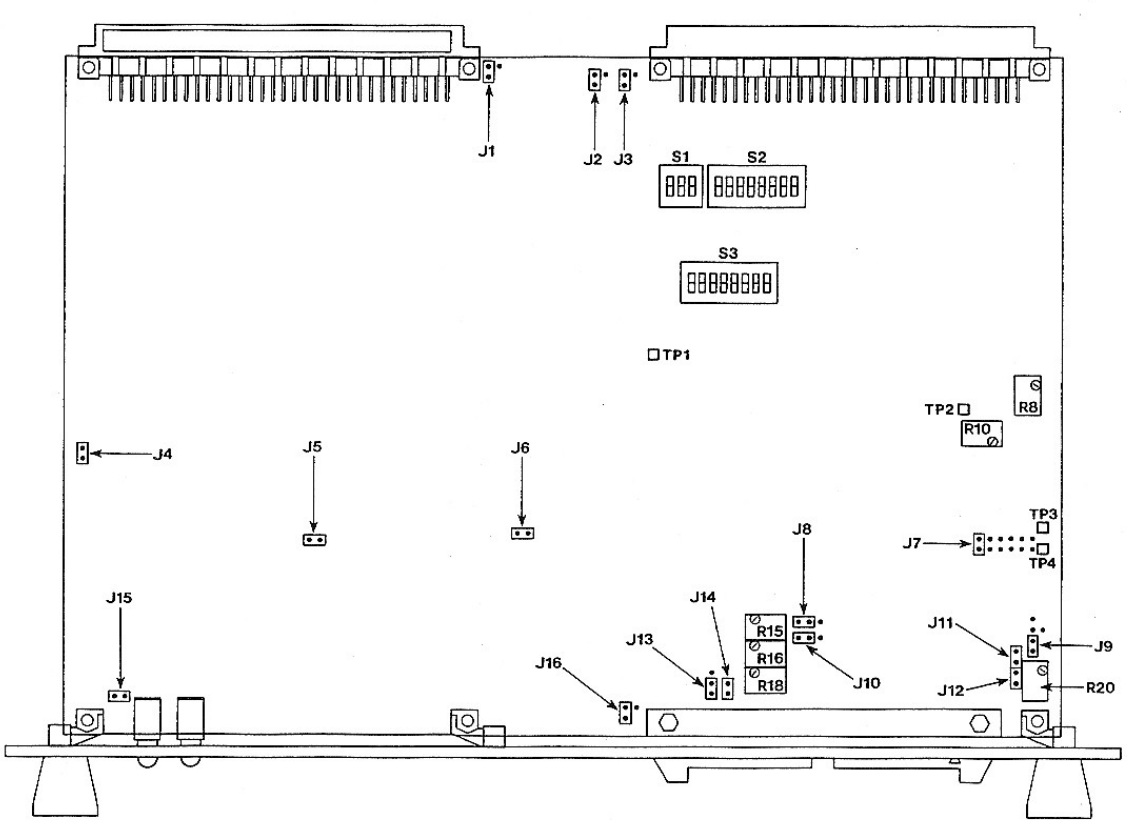


Figure 2-1

XVME-566 Jumpers, Switches, Test Points, Calibration Potentiometers and Connectors

Figure 2-2 provides a closer look at the module, depicting those jumpers with multiple options (more than two) and their relationship to the perimeters of the board.

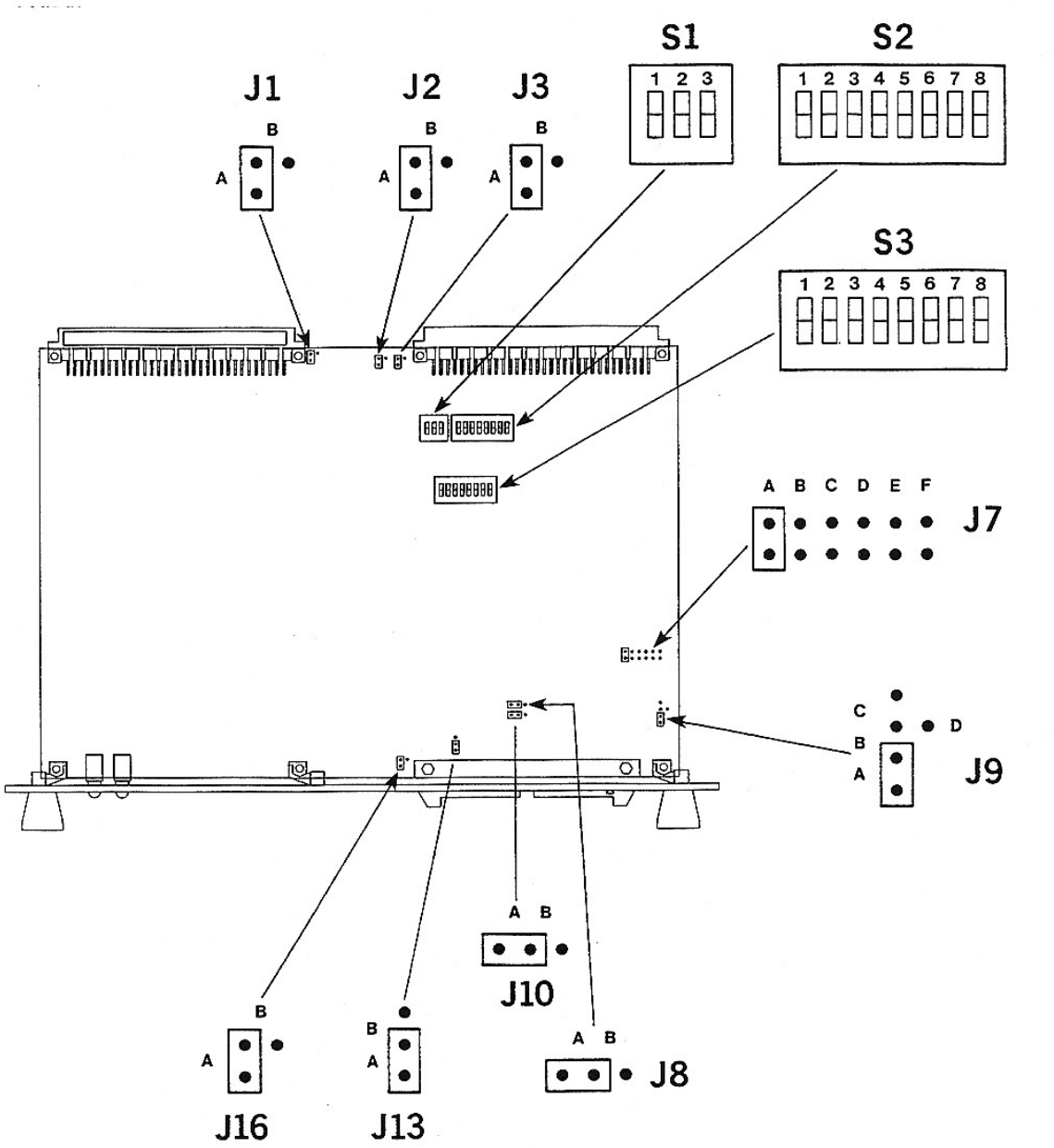


Figure 2-2

Jumpers and Switches with Multiple Options

2.4 JUMPERS

Prior to installing the XVME-566 module, several jumper and switch options must be configured. The configurations of the jumpers and switches are dependent upon the module capabilities required for a particular application. The jumper and switch options can be divided into two categories:

-- VMEbus Options, and
-- Analog-to-Digital (A/D) Conversion Options

Table 2-1 lists the various jumpers and their uses.

Table 2-1

XVME-566 Jumper Options

VMEbus OPTIONS	
Jumpers	**Use**
J1A,J1B,J2A,J2B	Enables, disenables IACKIN* to IACKOUT* daisy chain logic (See Section 2.6.4.2).
J3A,J3B	Standard or short I/O address select jumper (Section 2.6.2).

Analog-to-Digital Conversion OPTIONS	
Jumpers	**Use**
J4	Connects 8MHz clock to U54, Pin 1 for use with PAL device
J5,J16A,J16B	This group is used to select straight binary, offset binary and two's complement data format (Section 2.7.1).
J6	Configures data RAM size for 8K or 32K memory blocks (Section 2.6.1.2).
J7A,J7B,J7C,J7D,J7E,J7F	These jumpers are used to select one of three gain ranges for differential amplifier outputs (Section 2.7.4).

(continued on next page)

Table 2-1

XVME-566 Jumper Options (cont'd)

Analog-to-Digital Conversion Options (cont'd)	
Jumpers	**Uses**
J8A,J8B	Voltage range selector jumpers to analog-to-digital converter (Section 2.7.3).
J9A,J9B,J9C,J9D, J15	These jumpers provide the options of operating in single-ended, differential or pseudo-differential modes (Section 2.7.1.2).
J10A,J10B	These jumpers are used to configure inputs for either bipolar or unipolar input voltages and ranges (Section 2.7.3).
J11,J12,J14	These three jumpers are provided to allow for grounding of an input channel (in either the single[a]ended or differential mode) or for external ground reference for the external trigger (Sections 2.7.1.2, 2.7.2, 2.7.6).
J13A,J13B	Use of one of these jumpers will allow provisions for either on-board or off-board external triggers (Section 2.7.6).

2.5 SWITCHES

Several switch options must be configured before installing the Fast Analog Input Module. The configuration of the switches is dependent upon the modes and module capabilities required for the application.

Table 2-2 lists the various switches and their uses.

Table 2-2

XVME-566 Switch Options

Switch Bank S1	
Switches	**Use**
Switches 1-3	Interrupt level select for any interrupts generated by the module (Section 2.6.4.1).
Switch Bank S2	
Switches	**Use**
Switches 1-8	Data RAM base address select in standard address space (Section 2.6.1.1).
Switch Bank S3	
Switches	**Use**
Switches 1-6	Module I/O base address select (Section 2.6.1)
Switch 7	This switch determines whether the module operates with address modifiers for the short I/O address space, or for those within the standard address space (Section 2.6.3).
Switch 8	This switch determines whether the module will respond to only supervisory accesses, or to both supervisory and non-privileged accesses (See Section 2.6.3).

2.6 VMEbus OPTIONS

The following VMEbus options must be set up prior to installaiton of the XVME-566 module:

I/O Base Address
Data RAM Address and Size
Standard or Short I/O Selection for I/O Registers
Address Modifier Selection
VMEbus Interrupt Options

NOTE

Because the Data RAM can only occupy standard address space, individual base addresses must be set up for the Data RAM and the I/O registers (Gain RAM, Sequence RAM and STC registers). Switch Bank S3 places the I/O registers in short I/O or in the upper 64K of the standard memory space. Switch Bank S2 places the Data RAM in the standard memory space.

2.6.1 Module Base Address Selection Switches (S3-1 to S3-8)

The module base address is selected by using the switches labeled 1 through 8 in an eight-position DIP switch bank, S3. Figure 2-3 shows switch bank S3 and how the individual switches (1-8) relate to the base address bits (see Sections 2.6.2 & 2.6.3).

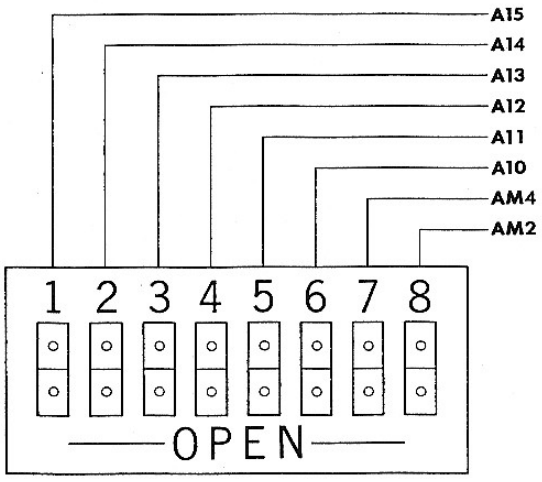


Figure 2-3

Switch Bank S3 - I/O Base Address Switches

When a switch is in the `closed' position (when set to the position opposite the "open" label), the corresponding base address bit will be logic `0'. When a switch is set in the `open' position, the corresponding base address bit will be logic `1'.

Table 2-3 shows a list of the 64 1K boundaries which can be used as module base addresses in both Short I/O Address Space and the Standard Memory Space. It also shows the corresponding switch settings (switches 1-8) from S3.

Table 2-3

Base Address Switch Options

Switches						VME base address in VME Short I/O Address space
1(A15)	2(A14)	3(A13)	4(A12)	5(A11)	6(A10)	
0	0	0	0	0	0	0000H
0	0	0	0	0	1	0400H
0	0	0	0	1	0	0800H
0	0	0	0	1	1	0C00H
0	0	0	1	0	0	1000H
0	0	0	1	0	1	1400H
0	0	0	1	1	0	1800H
0	0	0	1	1	1	1C00H
0	0	1	0	0	0	2000H
0	0	1	0	0	1	2400H
0	0	1	0	1	0	2800H
0	0	1	0	1	1	2C00H
0	0	1	1	0	0	3000H
0	0	1	1	0	1	3400H
0	0	1	1	1	0	3800H
0	0	1	1	1	1	3C00H
0	1	0	0	0	0	4000H
0	1	0	0	0	1	4400H
0	1	0	0	1	0	4800H
0	1	0	0	1	1	4C00H
0	1	0	1	0	0	5000H
0	1	0	1	0	1	5400H
0	1	0	1	1	0	5800H
0	1	0	1	1	1	5C00H
0	1	1	0	0	0	6000H
0	1	1	0	0	1	6400H
0	1	1	0	1	0	6800H
0	1	1	0	1	1	6C00H
0	1	1	1	0	0	7000H
0	1	1	1	0	1	7400H
0	1	1	1	1	0	7800H
0	1	1	1	1	1	7C00H
1	0	0	0	0	0	8000H
1	0	0	0	0	1	8400H
1	0	0	0	1	0	8800H
1	0	0	0	1	1	8C00H
1	0	0	1	0	0	9000H
1	0	0	1	0	1	9400H
1	0	0	1	1	0	9800H
1	0	0	1	1	1	9C00H
1	0	1	0	0	0	A000H
1	0	1	0	0	1	A400H
1	0	1	0	1	0	A800H
1	0	1	0	1	1	AC00H
1	0	1	1	0	0	B000H
1	0	1	1	0	1	B400H
1	0	1	1	1	0	B800H
1	0	1	1	1	1	BC00H
1	1	0	0	0	0	C000H
1	1	0	0	0	1	C400H
1	1	0	0	1	0	C800H
1	1	0	0	1	1	CC00H
1	1	0	1	0	0	D000H
1	1	0	1	0	1	D400H
1	1	0	1	1	0	D800H
1	1	0	1	1	1	DC00H
1	1	1	0	0	0	E000H
1	1	1	0	0	1	E400H
1	1	1	0	1	0	E800H
1	1	1	0	1	1	EC00H
1	1	1	1	0	0	F000H
1	1	1	1	0	1	F400H
1	1	1	1	1	0	F800H
1	1	1	1	1	1	FC00H

NOTE

Open = Logic "1"
Closed = Logic "0"

2.6.1.1 Data RAM Base Address Selection Switches (S2-1 to S2-8)

The Data RAM can only be accessed via the Standard Memory Space. It can be located anywhere in the 16-Mbyte address space on 64K ”word• boundaries.

The Data RAM base address is selected by using the switches labeled 1 through 8 in an eight-position DIP switch bank, S2. Figure 2-4 shows switch bank S2 and how the switches (1-8) relate to the base address bits. If a switch is open, the corresponding VMEbus address line must be active to enable access to the Data RAM.

E X A M P L E

To set Data RAM at base address A00000, the switches must be set as follows:

S2-1	S2-1	S2-3	S2-4	S2-5	S2-6	S2-7	S2-8
Open	Closed	Open	Closed	Closed	Closed	Closed	Closed

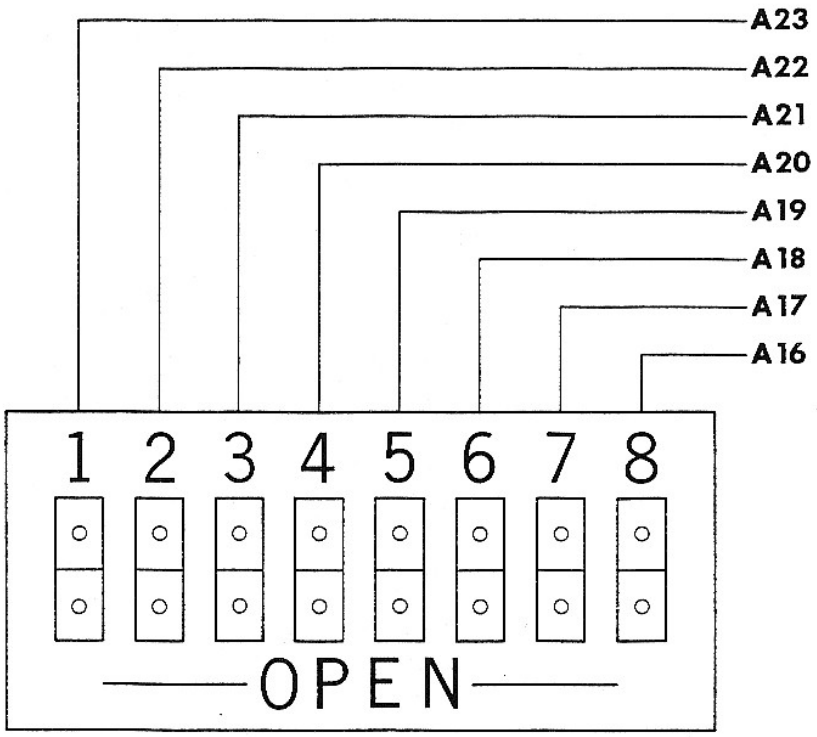


Figure 2-4

Data RAM Base Address Switches

Figure 2-4

Data RAM Base Address Switches

2.6.1.2 Data RAM Size Selection Jumper (J6)

Jumper J6 is used to configure the Data RAM for operation as a 32K word block, or an 8K word block. Installing J6, the module will run with a 32K word block of memory. If J6 is removed, the module will run with an 8K block of memory.

NOTE

Jumper J6 is shorted by a trace on the circuit side of the card. If the jumper is not used (that is: if the smaller, 8K, memory block are chosen), the trace must be cut. Figure 2-5 shows the location of the trace to be cut. There should be no need for the user to remove this jumper.

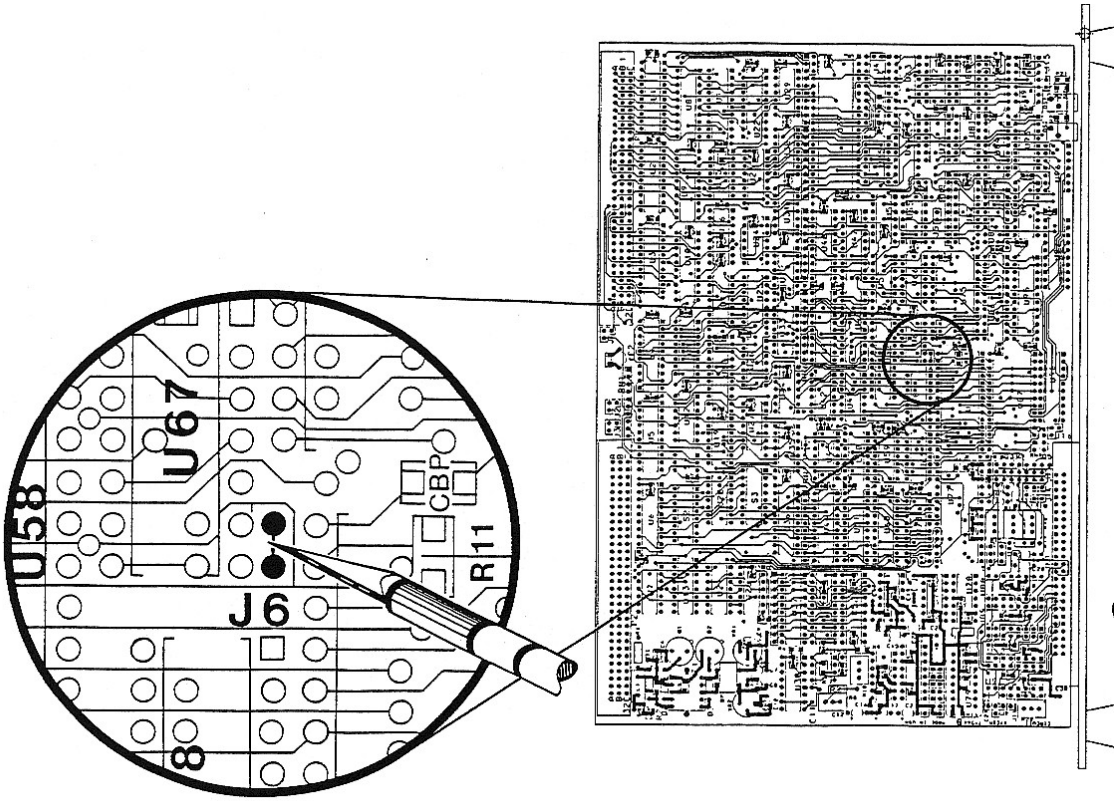


Figure 2-5

Modifying Data RAM for 8K Memory Block

Procedure

To modify the module for 8K memory block configuration:

1. Remove power from the module.
2. Remove module from the cardcage (continued next page).

3. Remove jumper J6 (if installed).

4. Locate the circuit trace shown in Figure 2-5.

5. Use a sharp tool (knife, X-acto blade, etc) to CAREFULLY sever the trace, as shown in Figure 2-5.

6. Re-install the module in the cardcage.

2.6.2 Standard or Short I/O Selection for I/O Registers (J3,S3)

All I/O module addresses (except the Data RAM) may be located in either VMEbus Short I/O or Standard Memory Space (the Data RAM can only be located in standard memory). The selection is made by configuring jumper J3 and switch 7 of Switch Bank S3 (see Figure 2-3) as shown in Table 2-4 below.

Table 2-4

I/O Module Addressing Modifier Options

Jumper	**Switch 7 (S3)**	**Option Selected**
J3A	Open	Standard Data Access Operation
J3B	Closed	Short I/O Access Operation

If jumper J3A is installed, S3-7 ”must be set in the OPEN position. If J3B is installed, S3-7 ”must be in the CLOSED position.

The XVME-566 is designed to take full advantage of the Standard I/O Architecture's various features (see Appendix A). It is therefore recommended that the module is operated in the Short I/O Address mode.

If the module is operated in the Standard Address Space, the FAIN Module will always reside within the upper 64K-byte segment of the Standard Memory Address Space (i.e., the address range FF0000H through FFFFFFH). Switches S3-1 to S3-6 then determine which 1K block of the upper 64K-byte segment will be occupied.

2.6.3 Address Modifier Selection (Supervisor/Non-Privileged Mode Selection)

The XVME-566 can be configured to respond only to Supervisory accesses, OR to both Supervisory and Non-Privileged accesses. The key is a combination of switch S3-7 and S3-8. Table 2-5 shows access options controlled by these switches, and the Address Modifier Code the module will actually respond to.

Table 2-5

Access Mode Options

Switch Set		**Address Modifier Code and Privilege Mode Selection**	
S3-8	**S3-7**	**I/O**	**Data RAM**
Closed	Closed	29H or 2DH Short I/O Supervisory Non-privileged Access	39H, 3DH Standard Supervisory Non-privileged Access
Closed	Open	39H, 3DH Standard Supervisory Non-privileged Access	39H, 3DH Standard Supervisory Non-privileged Access
Open	Closed	2DH Short I/O Supervisory	3DH Standard Supervisory
Open	Open	3DH Standard Supervisory Only	3DH Standard Supervisory Only

2.6.4 VMEbus Interrupt Options

Three interrupt switches (on Switch Bank 1) select which interrupt level is to be used by the module. The input module can be programmed to generate an interrupt at the completion of a conversion on any one of seven levels.

In order to enable interrupts, at least two bits in the Status/Control register must be set (See Chapter Four, Section 4.3.2.1). Interrupts may be initiated by the trigger clock (when sampling resumes after sequence is stopped by the Sequence RAM); by the Sequence RAM (when programmed and data is stored in the Data RAM); or by the event counter (when counter reaches programmed limit).

Interrupt resets must occur during the service routine. A logic `1' written to the upper control data bit 11 (timer), 13 (sequence), or 15 (event counter) will clear interrupts pending. Pending interrupts will also be cleared upon power-up or system reset.

2.6.4.1 Interrupt Level Switches (S1)

The following figure (Figure 2-6) shows Switch Bank 1 (S1). Table 2-6 (next page) defines the relationships between these switches and the module's interrupt levels.

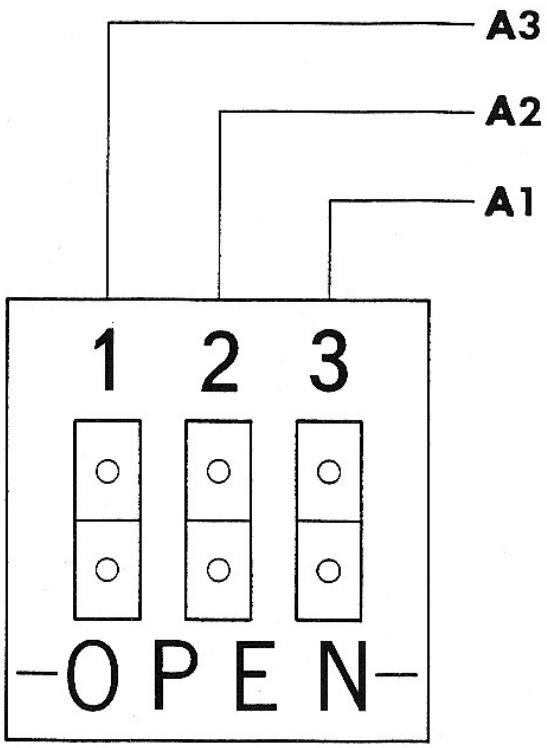


Figure 2-6

Interrupt Level Switches (S1)

The XVME-566 has the capability to generate interrupts on any one of seven levels (as allowed by VMEbus specification). Some of the possible sources for interrupts are: the Trigger Clock, Sample Clock (stop bit, interrupt bit), Event Counter, and Status/Control register (interrupt bits).

The module can be programmed to generate an interrupt at the completion of a conversion. These switches will determine the level of the interrupt. The interrupt level switch options are defined in Table 2-6.

Table 2-6

Interrupt Level Switch Options

(S1) Switches			VMEbus Interrupt Level
S1-1	**S1-2**	**S1-3**	
Open	Open	Open	7
Closed	Open	Open	6
Open	Closed	Open	5
Closed	Closed	Open	4
Open	Open	Closed	3
Closed	Open	Closed	2
Open	Closed	Closed	1
Closed	Closed	Closed	None, interrupts disabled

2.6.4.2 Interrupt Acknowledge (IACK*) Enable Jumpers (J1,J2)

The XVME-566 uses the VMEbus IACK* daisy-chain to determine which signal is acknowledged when more than one module uses the interrupt request lines. The daisy-chain through the module can be bypassed (to speed IACK* arbitration) when the module is not being used as an interrupter (but other modules are still free to interrupt). This is controlled by jumpers J1 and J2.

J1A connects IACKIN* to IACKOUT* at P1, and J1B enables IACKOUT* at the P1 connector. J2A disconnects IACKIN* from the module chain, while J2B enables IACKIN* to the module. Table 2-7 shows the interrupt acknowledge options.

Table 2-7

IACK* Enable Jumpers

Jumpers		Option
J1	**J2**	
Position A	Position A	Module bypasses IACK* daisy chain
Position B	Position B	Module uses IACK* daisy chain

When the jumpers are in the `A' position, the module will not respond to interrupts because IACKIN* is not monitored (the on-board arbitration is not engaged). The `A' position connects IACKIN* directly to IACKOUT*, at the P1 connector.

The `B' position engages IACKIN* to the on-board arbitration circuitry. If interrupts are to be used, these jumpers should be in the `B' position.

2.7 ANALOG-TO-DIGITAL (A/D) CONVERSION OPTIONS

2.7.1 Input Conversion Format Jumpers (J5,J16)

This jumper option is used to configure A/D conversion circuitry to convert analog information to one of three formats: straight binary (unipolar), offset binary (bipolar), or two's complement binary (bipolar).

Use of this option is dependent upon the data format required by the input control program employed by the user. This option is inclusive to all input channels and cannot be utilized on an individual channel basis (Section 2.7.1.1).

The digital-data format stored into the Data RAM may be changed (via jumpers) to accommodate several types of data encoding. Table 2-8 shows which jumpers control the data formats for the two different voltage modes.

Table 2-8

Input Conversion Format Jumpers

Digital Data Conversion Format (All Inputs)	Jumpers Installed	Input Mode
Analog to Straight Binary	J5,J16B	Unipolar
Analog to Offset Binary	J5 (Out), J16B	Bipolar
Analog to Two's Complement	J5,J16A	Bipolar

2.7.1.1 A/D Data Format

The A/D converter digitizes the value of an analog signal on the input of a selected channel. The digital format of the converted data depends upon which data format and input voltage mode (unipolar or bipolar) have been previously jumpered at module installation (Section 2.7.1).

The analog input signals can be divided into two general groups: unipolar input, where the input has only positive polarity (e.g., 0-5V or 0-10V); and bipolar input, where the input magnitude can "swing" between a positive and a negative polarity (e.g., -5V to +5V or -10V to + 10V).

If the inputs are configured to accept unipolar voltages, the straight binary format of data coding is usually selected. If the inputs are configured to accept bipolar voltages, the data can be encoded in either offset or two's complement (to encompass handling negative numbers). The three formats are listed in the following three tables. Table 2-9 shows the straight binary encoding format.

Table 2-9 Unipolar Mode
(Straight Binary Encoding)

Bits D15 through D12 always = zero

Straight Binary: (Vfsr = Full Scale)																	
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Analog Input**
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	Vfsr - 1LSB 0FFFH
	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	.5 Vfsr 0800H
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0V 0000H

In the bipolar mode, the analog value converted to digital is encoded in either Offset Binary or Two's Complement form.

In Offset Binary, the negative full scale voltage (-Vmax) is represented by all binary zeros. The positive full scale voltage (Vfsr-1LSB) is represented by all binary ones. Thus, the voltage represented is 'offset' by a factor of one half the full scale voltage "swing" (+Vmax to -Vmax). Table 2-10 shows the data encoding format for the bipolar mode with offset binary encoding.

Table 2-10 Bipolar Mode
(Offset Binary Encoding)

Bits D15 through D12 always = zero

	Offset Binary:																**(Vfsr = Full Scale)**	
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Analog Input**	
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	Vfsr - 1LSB	0FFFH
	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	.5(+1Vfsr)	0C01H
	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0V	0800H
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	.5(-1Vfsr)	0400H
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-Vfsr	0000H

In the Two's Complement mode, the most significant bit is simply inverted. This provides for direct mapping between the Two's Complement numbers used by the microprocessor and the voltage input of the analog-to-digital converter. Table 2-11 shows the (bipolar mode) two's complement encoding format.

Table 2-11 Bipolar Mode
(Two's Complement Encoding)

** Bits D15 through D12 always match Bit D11 **

	Two's Complement:																**(Vfsr = Full Scale)**	
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Analog Input**	
	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	Vfsr - 1LSB	07FFH
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	.5(+1Vfsr)	0400H
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0V	0000H
	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	.5(-1Vfsr)	FC00H
	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	-Vfsr	F800H

2.7.1.2 Differential/Pseudo-Differential/Single-Ended Input Jumpers (Analog Multiplexers J9,J14,J15)

The XVME-566 can be configured to provide any one of three types of channel input modes:

1) 32 single-ended (SE) input channels

2) 16 differential (DI) input channels, or

3) 32 pseudo-differential (PDI) input channels (allows module to simulate advantages of true differential with some degradation)

The three modes are mutually exclusive, so the module will only accept all SE, or all DI, or all PDI inputs. It will not accept combinations of the three.

NOTE

Only one of the three jumper configurations may be installed at any one time. Make sure that:

1) If the board is to be used in the SE input mode, the jumpers for DI and PDI operation are removed;

2) If the board is to be used in the DI input mode, the jumpers for SE and PDI operation are removed;

3) If the board is to be used in the PDI input mode, the jumpers for SE and DI operation are removed.

The pseudo-differential option allows 32 input channels, as does the SE option. Unlike the SE channel, the PDI mode uses a common analog ground (pin JK1-49) to simulate true differential input (IMPORTANT: see Section 2.7.2). Table 2-12 shows how jumpers J9,J14 and J15 must be used to achieve SE, DI and PDI inputs.

Table 2-12

Single-Ended/Differential/Pseudo-Differential Jumper Options

Jumper Settings			**Input Mode**
J9	**J14**	**J15**	
A,C	Out	In	SE
B	Out	Out	DI
A,D	Out	In	PDI

NOTE

The LSB (Least Significant Bit) represents the change in input voltage that results in an increase or decrease of the binary code by one count. The LSB is derived from the full range of either current or voltage (Vfsr), divided by the maximum conversion resolution (i.e., 12 bits or 4096 in binary counts). Thus, the value of one LSB can be determined by the following:

$$\text{Unipolar LSB} = \frac{\text{Vfsr}}{4096} \qquad \text{Bipolar LSB} = \frac{(+\text{Vfsr})-(-\text{Vfsr})}{4096}$$

The following list shows the value of 1 LSB for each range:

± 5V	=	2.4414mV
± 10V	=	4.8828mV
0 - 10V	=	2.4414mV

2.7.2 Input Calibration Grounding Jumpers (J11,J12)

In addition to the SE/DI jumpers mentioned, two other jumpers are provided to allow grounding of an input channel (in either the SE or DI mode). This is done to allow software to automatically correct any drift in the ADC offset adjustment.

In the SE mode, J11 grounds input CH.8 and J12 grounds input CH.0. In the DI mode, J11 grounds CH.O HI and J12 grounds CH.O LO (See JK1 Table, SEC. 2.8 for reference). These jumpers will only be installed during the calibration procedure (Chapter 5). Table 2-13 shows the effects of grounding these jumpers.

Table 2-13

Inputs Grounded on JK1

Jumper	Mode	Result
J11	SE,PDI	CH. 8 grounded
J12	SE,PDI	CH. 0 grounded
J11	DI	CH. 0 HI grounded
J12	DI	CH. 0 LO grounded

2.7.3 Input Voltage Range Type and Voltage Range Selection (J8,J10)

The analog inputs may be configured to accept either unipolar or bipolar full-scale input voltages. Jumpers J8 & J10 determine which voltage type the module will accept.

The analog input ranges can be jumper-configured to accept voltages in any one of three ranges. There are two bipolar ranges and one unipolar range:

Bipolar Ranges	Unipolar Range
± 5V	0 to 10V
± 10V	

Table 2-14 shows the voltage range and type options.

Table 2-14

Voltage Range and Type Selection Options

Input Range	Install
Unipolar:	
0-10V	J8A,J10A
Bipolar:	
± 5V	J8A,J10B
± 10V	J8B,J10B

2.7.4 Programmable Gain Selection (J7-Jumper Cluster)

The gain for each input channel is programmable over any of three possible gain ranges. First, the required range is selected by configuring one of the three jumper-pair combinations for jumper J7. Next, the specific gains (within the selected range) are determined by the user and written to the on-board Gain RAM during the input initialization procedure (Section 4.4.1).

Anytime a subsequent input is converted (analog-to-digital), the gain factor will automatically be applied as it was previously programmed. The three input gain ranges are:

Three Input Gain Ranges

Range 1:(X1)	1,2,5,10
Range 2:(X4)	4,8,20,40
Range 3:(X10)	10,20,50,100

The various input gains are selected by installing one jumper pair for each option. Table 2-15 shows the options and corresponding jumper pairs.

Table 2-15

Input Gain Range Jumper Pairs

Jumpers Installed	**Gain Range Selected**
J7A,J7B	Range 1
J7C,J7D	Range 2
J7E,J7F	Range 3

SELECT ONLY ONE RANGE AT A TIME. The input channels can only be programmed for specific gains within the selected range.

2.7.5 Conversion Resolution

The XVME-566 is capable of performing with the ADC in an 8-bit or a 12-bit conversion mode. The module powers up in the 12-bit mode. (Refer to Chp 4, Status/Control register for selection of 8-bit option.)

The 8-bit option allows a faster conversion (up to 144KHz throughput, if required by the user), but will decrease conversion accuracy. When the 8-bit conversion is chosen, the conversion will still have 12-bit resolution, BUT will have 8-bit accuracy. The system accuracy for 12-bit and 8-bit conversions, respectively, are ± 0.01% FSR max., and ± 0.2% FSR max.

2.7.6 External Trigger Selection (J13,J14)

On the XVME-566,jumper J13A is installed to connect an external trigger input to the module. When J14 is installed, pin JK1-49 becomes the logic ground reference. (When J14 is installed, jumper J9D must be removed.) The PDI mode cannot be used with the external trigger.

Jumper J13B is used to provide external trigger output for the XVME-566 at pin number JK1-50.

2.8 CONNECTOR JK1

The analog input channels are accessible on the front of the module via the single mass termination header labeled, JK1. Connector JK1 is a 50-pin header used to input analog signals. Its pin-out is compatible with the Analog Devices 3B series Universal Signal Conditioning System. Table 2-16 defines JK1's pin-out.Ü"□□□□□□Ü

Table 2-16 Input Connector JK1

Flat Cable Conductor	**Single-Ended Configuration**	**Differential Configuration**
1	CH. 0	CH. 0 LO
2	CH. 8	CH. 0 HI
3	ANALOG GND	ANALOG GND
4	CH. 9	CH. 1 HI
5	CH. 1	CH. 1 LO
6	ANALOG GND	ANALOG GND
7	CH. 2	CH. 2 LO
8	CH. 10	CH. 2 HI
9	ANALOG GND	ANALOG GND
10	CH. 11	CH. 3 HI
11	CH. 3	CH. 3 LO
12	ANALOG GND	ANALOG GND
13	CH. 4	CH. 4 LO
14	CH. 12	CH. 4 HI
15	ANALOG GND	ANALOG GND
16	CH. 5	CH. 5 HI
17	CH. 13	CH. 5 LO
18	ANALOG GND	ANALOG GND
19	CH. 6	CH. 6 LO
20	CH. 14	CH. 6 HI
21	ANALOG GND	ANALOG GND
22	CH. 15	CH. 7 HI
23	CH. 7	CH. 7 LO
24	ANALOG GND	ANALOG GND
25	CH. 16	CH. 8 LO
26	CH. 24	CH. 8 HI
27	ANALOG GND	ANALOG GND
28	CH. 25	CH. 9 HI
29	CH. 17	CH. 9 LO
30	ANALOG GND	ANALOG GND
31	CH. 18	CH. 10 LO
32	CH. 26	CH. 10 HI
33	ANALOG GND	ANALOG GND
34	CH. 27	CH. 11 HI
35	CH. 19	CH. 11 LO
36	ANALOG GND	ANALOG GND
37	CH. 20	CH. 12 LO
38	CH. 28	CH. 12 HI
39	ANALOG GND	ANALOG GND

(continued on next page)

Table 2-16

Input Connector JK1 (cont'd)

Flat Cable Conductor	Single-Ended Configuration	Differential Configuration
40	CH. 29	CH. 13 HI
41	CH. 21	CH. 13 LO
42	ANALOG GND	ANALOG GND
43	CH. 22	CH. 14 LO
44	CH. 30	CH. 14 HI
45	ANALOG GND	ANALOG GND
46	CH. 31	CH. 15 HI
47	CH. 23	CH. 15 LO
48	ANALOG GND	ANALOG GND
49	GND (EXT TRIG,PDI)	GND (EXT TRIG,PDI)
50	EXT TRIGGER	EXT TRIGGER

NOTE
When performing conversions in Sequence Mode, a software reset will lock the board. Before doing a software reset in this mode, bit 8 of the Status/Control register must be set to "0" (Sequence Controller Enable).

2.9 JUMPER LIST

The following table summarizes all the XVME-566 jumpers and their functions.

Table 2-17

XVME-566 Jumper List

Jumper	**Description**
J1A	Connects IACKIN* to IACKOUT* at P1
J1B	Engages IACKIN* to IACKOUT* to daisy chain circuitry
J2A	Disconnects IACKIN* from daisy chain circuitry
J2B	Enables IACKIN* to daisy chain circuitry
J3A	Enables module for standard memory access
J3B	Enables module for short I/O access
J4	Jumpers 8 MHz clock to U54, Pin 1
J5	Analog-to-straight binary conversion
J6	Configures Data RAM for 32K memory blocks
J7A	Differential Amp output Stage Gain of X1
J7B	Differential Amp output Stage Gain of X1
J7C	Differential Amp output Stage Gain of X4
J7D	Differential Amp output Stage Gain of X4
J7E	Differential Amp output Stage Gain of X10
J7F	Differential Amp output Stage Gain of X10
J8A	Voltage range selector to ADC; ”+•10V
J8B	Voltage range selector to ADC; ”+•20V
J9A	Configures module for SE & PDI operation
J9B	Configures module for DI operation
J9C	Configures module for SE operation; accompanies J9A
J9D	Configures module for PDI operation; accompanies J9A
J10A	Unipolar voltage range selector with potentiometer R18
J10B	Bipolar voltage range selector with potentiometer R16
J11	Grounds JK1 SE CH.8 and DI CH.0 HI
J12	Grounds JK1 SE CH.0 and DI CH.0 LO
J13A	External trigger selection provision
J13B	Off-board trigger selection provision
J14	Grounds JK1 Pin 49 for external trigger reference
J15	Configures module for SE operation; accompaniesJ9A
J16A	Sign extends MSB (BD11) of converted result (with J5,selects straight binary; without J5, selects offset binary)
J16B	Inverts MSB (BD11) of converted result (with J5, selects two's complement)

2.10 POTENTIOMETER/RESISTOR LIST

The following table summarizes the XVME-566 potentiometers and their functions.

Table 2-18

XVME-566 Potentiometer List

Resistor Number	Description
R8	One-shot time out
R10	Sample/hold offset adjustment
R15	Gain adjustment, ADC
R16	Bipolar offset adjustment, ADC
R18	Unipolar offset adjustment, ADC
R20	Instrumentation offset adjustment

2.10.1 Current Loop Inputs

An A/D input will operate in a 4-20mA or a 10-50mA current loop configuration with the addition of an external current sensing resistor. The current sensing resistor should be selected to generate a voltage within the predetermined, jumper-selected voltage (0-5V max.). A voltage drop of less than 1V will provide current of less than 4mA, and would thus indicate improper operation of the current loop.

Typically, the resistors used would be:

A 500-ohm 1/2W for the 4-20mA configuration,

OR

A 200-ohm 1/2W for the 20-50mA configuration.

The resistors should be 0.1% tolerance or better, with stable temperature coefficient characteristics (e.g., 25ppm or better). All input channels operate with the same full-scale input range.

2.11 MODULE INSTALLATION

XYCOM XVME modules are designed to comply with all physical and electrical VMEbus backplane specifications. The XVME-566 Fast Analog Input Module is a double-high VMEbus module. Two backplane connectors are installed (P1 and P2). The module requires P2 for power and ground.

CAUTION

Never attempt to install or remove any module before turning the power to the bus OFF. Power to all related external power supplies should also be OFF.

Prior to installing the module, determine and verify all relevant jumper configurations. Check all connections to external devices or power supplies to make sure they comply with the specifications of the module. (Please check the jumper configuration against the diagrams and lists in this manual.)

To install the module into the cardcage, perform the following steps:

1) Make sure the cardcage slot (which will hold the module) is clear and accessible.

2) Center the module on the plastic guides so the solder side is facing the left and the component side is facing right.

3) Push the card slowly toward the rear of the chassis, until the connectors engage (the card should slide freely in the plastic guides).

4) Apply straightforward pressure to the handles on the front panel, until the connector is fully engaged and properly seated.

NOTE

It should not be necessary to use excessive force or pressure to engage the connectors. If the board does not properly connect with the backplane, remove the module. Then inspect all connectors and guide slots for possible damage or obstructions.

5) Once the module is properly seated, secure it to the chassis by tightening the two machine screws at the extreme top and bottom of the module.

Chapter 3

PROGRAMMING OVERVIEW

3.1 INTRODUCTION

Use of the XVME-566 is simplified with a general understanding of its three major spheres: the RAMs, the on-board System Timing Controller and the operating modes. This chapter discusses these module components in the following order:

3.2 THE RAMS

The XVME-566 uses three RAM memory structures. There is one each for data storage, sequence-of-events programming, and gain-level programming. All three RAM areas are generally used when the board is operated.

3.2.1 The Gain RAM (Base + 101H)

On the XVME-566, the Gain RAM is used to store the gain values for the channels whose data will be converted. The 32-byte RAM occupies the odd bytes in the I/O address space from base address + 101H through base address + 13FH.

The Gain RAM is configured so that only the two least significant bits are used. Within those bits, gains of 1,2,5 or 10 are stored within three jumper-selectable gain ranges. Section 2.7.5 discusses the gain ranges in more detail. Section 4.3.4.1 lists the procedures for using the Gain RAM.

3.2.2 The Sequence RAM (Base + 201H)

The 256-byte Sequence RAM is the heart of the Sequence Controller. It is used to program the order in which data from input channels are converted. The RAM occupies the odd bytes in the I/O address space from base address + 201H through base address + 3FFH.

All bits in this RAM are used to manipulate data. The five least significant bits are used to store input channel addresses. The remaining three bits are control bits:

-- End-of-Sequence Bit: If the IMAGE bit is set in the status control register, this bit will cause the sequence and data RAM pointers to reload. If the IMAGE bit is not set, only the Sequence RAM pointer will reload.

-- Stop Bit: (Sequential Mode only) After a conversion is stored, this bit causes the sequence to stop

-- Interrupt Bit: After data is stored, this bit causes an interrupt to the VMEbus (if enabled)

Section 4.3.4.4 discusses the Sequence RAM in more detail.

3.2.3 The Data RAM

The XVME-566 uses a 64Kbyte, dual-ported Data RAM to store converted data samples (up to 32,767 12-bit samples). Access to the RAM contents is provided by pointers in the Data RAM Address Register (base + word location 84H).

The Data RAM can be located anywhere in the 16Mbyte Standard Memory Space on 64K byte boundaries. The size of the RAM can be either 8K or 32K words (the board is shipped with 32K words). Section 2.6.1.1 discusses selection of the Data RAM address space. Details regarding the use of the RAM can be found in Section 4.3.4.2.

The following sections discuss the two data storage modes.

3.2.3.1 Data Storage Modes

The module is designed to allow storage of data in one of two modes: image and contiguous. When power-up occurs, the board is set for the the contiguous mode. Changing the setting of a bit in the status/control register will invoke the image mode.

The next two sections describe the contiguous and image data storage modes in greater detail.

3.2.3.1.1 Contiguous Mode

The contiguous mode provides a history of all recorded values at specified channels. Contiguous mode is already set in the status/control register at power-up. Otherwise, it is invoked by setting bit 6 in the status/control register to logic`0'.

Contiguous sequencing differs from image sequencing where the storage of data is concerned. When in the contiguous mode, data is stored from a start address (specified by the Data RAM pointer), permitting a series of input data values to be stored for each channel. The result is the creation of a block of Data RAM that contains a history of the events that have occurred on the channel.

The "end-of-sequence" bit in the Sequence RAM will not cause the Data RAM counter to be reloaded (with its initial value) when the board is operating in the contiguous mode. Thus, data is continually stored in successive locations in the Data RAM until the stop bit is set (stopping the acquisition of data).

Example

Fifteen channels represent the plot points for a sine wave being produced by an oscillator. Company regulations require that a continuous record of these points be recorded every five minutes during the equipment's twenty-hour operating time each day. Contiguous mode allows the status (of the fifteen points at separate addresses in the Data RAM) to be recorded after each conversion is made. The time in this example is controlled by the STC trigger clock.

3.2.3.1.2 Image Mode

The image mode is designed to provide the latest information available on a given channel. Data is stored in an assigned area of the Data RAM (as determined by Data RAM pointer). The area may then either be read once or continually updated for a selected group of input channels. the image mode is invoked by setting bit 6 in the status/control register to logic `1'.

When in the image mode, the Data RAM pointer will be loaded with the initial value whenever the end of the sequence is reached. This allows the new data to overwrite the old data (making each store to the Data RAM the latest value).

Example

Acquisitions are needed every fifteen minutes on equipment that is controlled by five channels assigned to Work Station Nineteen. The image mode allows data from each of the channels to be acquired in a single routine. So, every fifteen minutes, the converted values for the five channels are acquired then stored in the Data RAM. Because the purpose of the routine is to collect updated data values only, new values are stored over the old values.

3.3 SYSTEM TIMING CONTROLLER FUNCTIONS

The System Timing Controller (STC) on the XVME-566 is a versatile source for coordinating timing sequences. Uses for the STC are expansive. This manual, however, will discuss only those features which are dedicated to specific counter outputs. More detailed information, if necessary, can be found in the STC handbook provided with the module.

The XVME-566 uses the Am9513A STC which includes five 16-bit counters and a four-bit prescaler. Each counter may also be concatenated to any succeeding counter for an effective counter length of 80 bits.

The primary STC features used by the XVME-566 are:

--	Counter 4:	The Sample Clock
--	Counter 2:	The Trigger Clock, and
--	Counter 5:	The Event Counter

Some examples for programming the STC can be found in Chapter 4 of this manual. The following sections briefly discuss the sample clock, trigger clock and event counter features.

3.3.1 The Sample Clock

The Sample Clock determines the time between each acquired bit of data in a particular sequence of events. When active, the sample clock allows the sampling of data to occur (in the length of time specified), in the specified sequence, until the sequence has ended.

When the sequence has ended (when EOS or STOP are set in Sequence RAM), the sample clock will not resume counting until it is re-started by the trigger clock, a software trigger or an external trigger. The Sample Clock is provided by output 4 of the STC.

In Sequential Mode, the sample clock controls the throughput rate of data acquisition. It limits the sampling rate for 8-bit conversions to a minimum of seven microseconds (144KHz). Twelve-bit conversions cannot be faster than ten microseconds. The desired sampling rate only needs to be set once (after power-up), unless re-programming is necessary for other functions.

NOTE

Upon initialization, the sample clock Terminal Count (TC) must be low for 500nSec. Any deviation will cause unpredictable results. (Terminal Count is defined as: that period of time when the counter contents would have been zero had an external value not been transferred into the counter.)

3.3.2 The Trigger Clock

The Trigger Clock occupies Output 2 of the STC. The purpose of this trigger is to control the time between sequences by starting or re-starting the sampling process.

NOTE
The trigger clock period must always be greater than the total sampling period.

Example

In the diagram below (Figure 3-1), the sample clock periods are represented as shown with thresholds occurring every ten microseconds (programmed after power-up). Because the sample clock periods are set, however, does not mean that sampling is actually occurring.

In Periods A & C, no sampling occurs. Sampling does not occur in Period A because no trigger has "allowed" the sampling process to begin. The first trigger occurs at the start of Period B. The sampling stops where the setting of the "stop bit" is indicated. Period C (all of the time remaining after the stop bit) contains no new trigger, so no new sampling occurs. At the beginning of Period D, however, a new trigger occurs and sampling resumes.

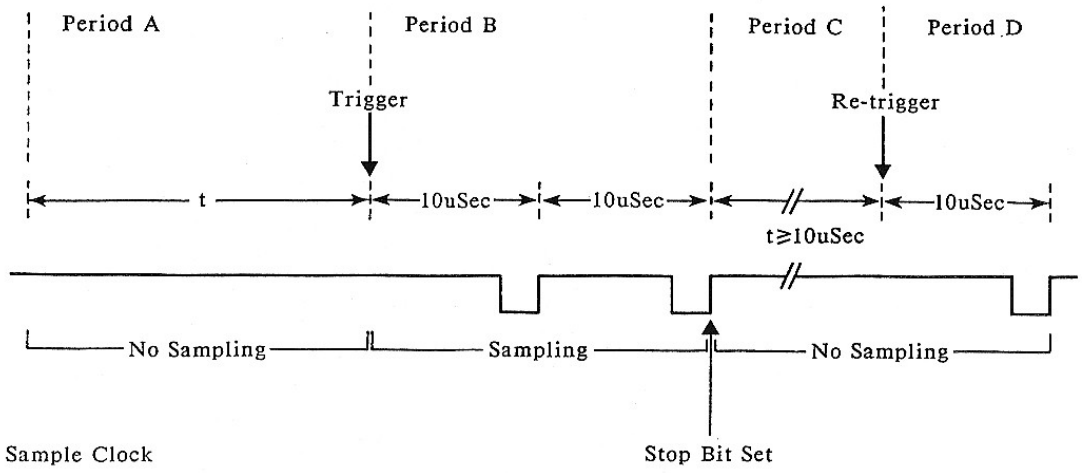


Figure 3-1

Triggering the Sampling Process

3.3.3 The Event Counter

The Event Counter occupies Output 5 on the STC. Its primary function is to count "events" and provide an interrupt to the VMEbus. Events can be:

-- The number of times the trigger acts (STC input G4), or
-- The number of sample clock periods (STC input S4), or
-- The number of end-of-sequence settings (STC input G3), or
-- The number of external triggers generated (STC input G1)

3.4 ANALOG CONTROL (THE SEQUENCE CONTROLLER)

Once initialization has occurred, the XVME-566 operation is coordinated by a sequence controller that accesses the Gain, Data and Sequence RAMs, requiring trigger and sample clock inputs. The RAMs, trigger and sample clocks are discussed in Sections 3.2 and 3.3.

The Sequence Controller coordinates the actions between the analog components, the RAMs and their respective pointers. Along with the STC the sequence controller provides the proper timing for data acquisition and dual S/H amplifiers switching.

Timing for transferring data into the Data RAM (at the end of each conversion) is also controlled by the sequence controller. Finally, the controller provides the arbitration necessary for VME access to any of the RAMs during the acquisition process.

3.5 OPERATING MODES

The XVME-566 operates in one of two modes, Sequential or Random. The more versatile Sequential Mode will almost always be the preferred mode of operation because it takes full advantage of the board's speed and storage capacities.

Sequential mode operations use the dual sample-and-hold architecture (which allows data on one channel to be held while another channel is being sampled). Random mode operations allow acquisition and conversion from one channel at a time (which slows the throughput rate).

Both modes allow data acquisition in 10uSec and expend an additional 10uSec converting and storing data. The Sequential mode, however, performs both tasks simultaneously (except on the first acquisition and the last conversion).

The remainder of this chapter discusses the Sequential and Random modes. Table 3-1 shows the registers that must be initialized prior to data acquisition in either mode.

Table 3-1

Registers Initialized Before Data Acquisition

Register	Mode	
	Sequential	**Random**
System Timing Controller	Yes	Yes
Sequence RAM	Yes	No
Gain RAM	Yes	Yes
Sequence RAM Pointer	Yes	No
Data RAM Pointer	Yes	Yes
Interrupt Vector	Yes	Yes
Status/Control Register	Yes	Yes

3.5.1 Sequential Mode

In the sequential mode, conversions are performed at a rate of 100KHz (maximum with 12-bit accuracy). This is accomplished with a dual sample/hold architecture that allows data acquisition and conversion to occur simultaneously. The sampling sequence is controlled by a Sequence RAM which permits greater flexibility such as the looping of sample sequences, VMEbus interruption at sample completion, and termination of the sampling. The sequence of samples may be initiated by:

1. Trigger Clock
2. External Trigger
3. Software Trigger

The sequential mode allows data to be sampled (acquired, read and stored) on many channels in a single command set. Channel numbers and control data are retrieved from the 256-byte Sequence RAM (before conversion) and converted data is stored in the 32K word Data Ram. The Sequential Mode is selected by setting bit 5 in the status/control register to logic`0' (the module powers-up in this mode).

Data can be stored in one of two ways, in the Image or Contiguous mode. When only the latest information on specified channels is needed, the Image Mode is used. The second type, Contiguous Mode, is used for plotting a history of events at the specified locations. See section 3.2.3.1 for more information on the data storage modes.

Figure 3-2 shows a time line for sample acquisition in the Sequential mode.

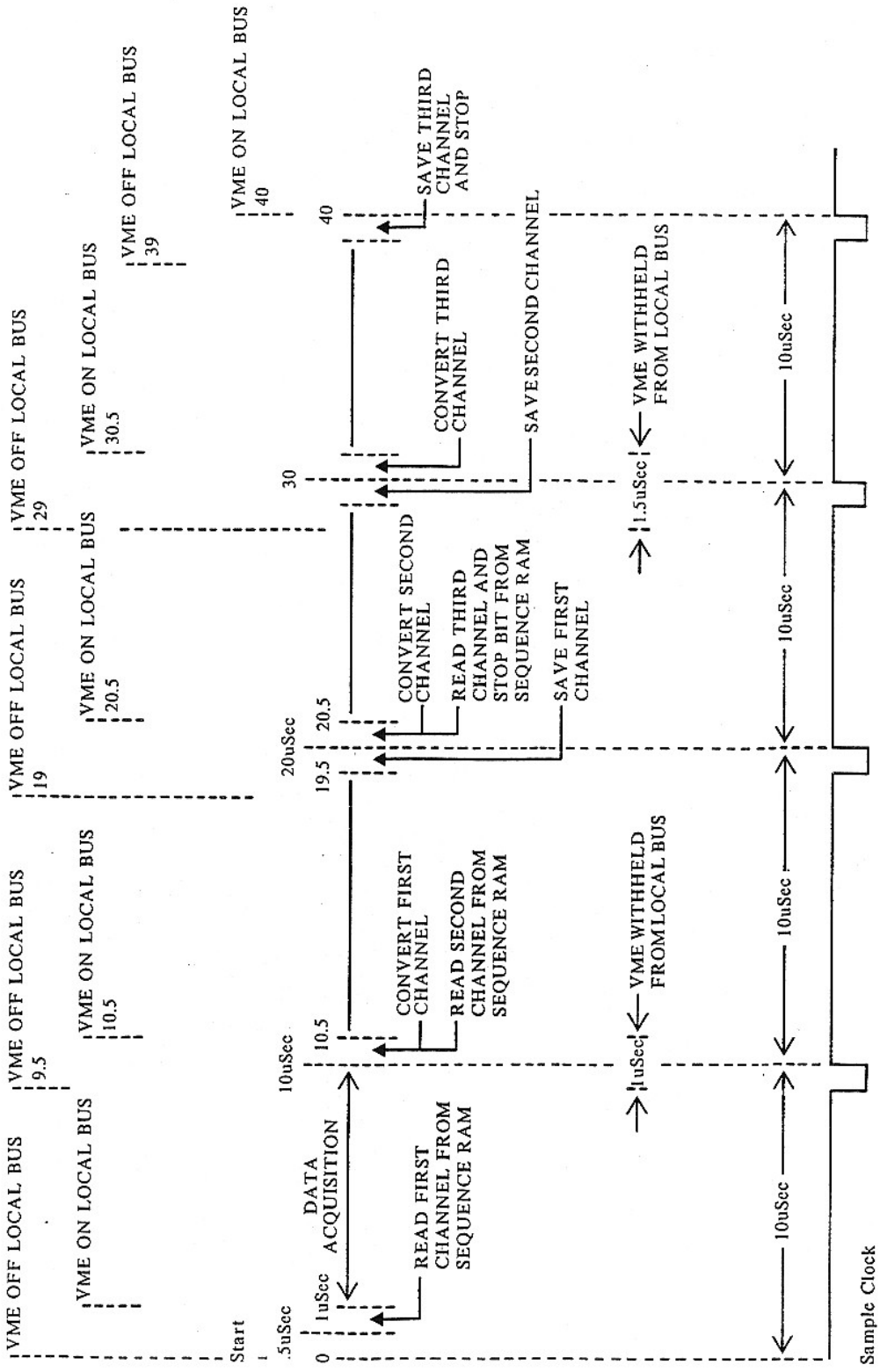


Figure 3-2

Time Line For Sequential Mode Data Acquisition

3.5.2 Random Mode

In the random mode, a control byte is written to the Sequence RAM or the Gain RAM (specifying a channel number) to start a conversion on the channel. Random mode operation is accessed by setting bit 5 of the status/control register to logic `1'.

Sequence RAM and Gain RAM values do not change when in the Random mode. The write to the RAM causes a latch to be loaded instead. Random mode uses both the image and contiguous data storage modes. See section 3.2.3.1 for more information on data storage.

Other trigger modes (External Trigger, Trigger Clock, Software Trigger) should not be used to start data acquisition in the random mode. They may be used, however, to continue data acquisition and conversion in this mode. When restarting the acquisition process via one of the three triggers, the data written to the multiplexer is not changed. Therefore, when a trigger occurs, another conversion will take place on the channel initially written.

Figure 3-3 shows a time line for data acquisition and conversion in the Random mode. See Figure 3-2 for comparison with the Sequential mode.

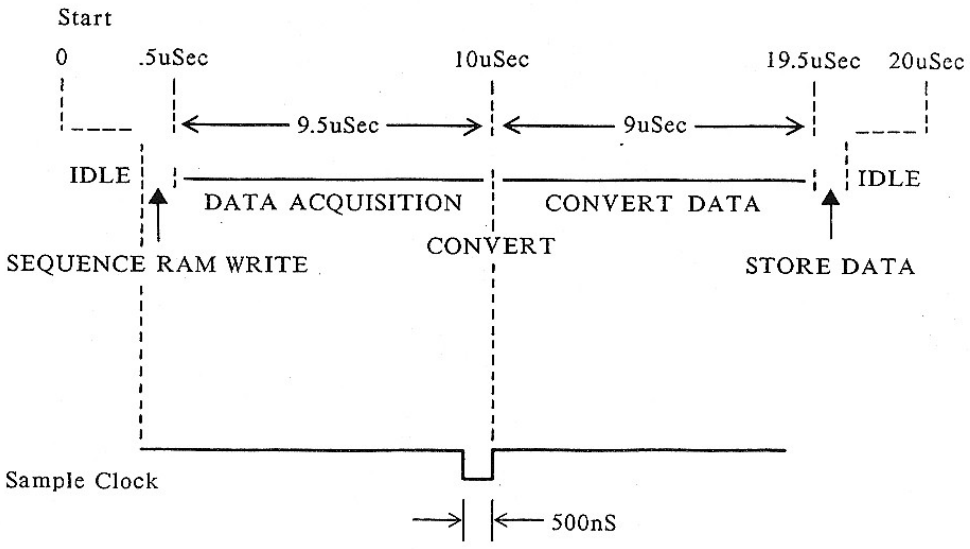


Figure 3-3

Time Line for Data Acquisition
Random Mode

Chapter 4

PROGRAMMING

4.1 INTRODUCTION

This chapter provides information required to program the XVME-566 Fast Analog Input Module for analog-to-digital conversions. Most of the information for programming the System Timing Controller can be found in the STC manual accompanying the module. The presentation of information in this chapter is as follows:

- Base address and module identification locations
- Discussions of base addressing and how the various registers are accessed
- Discussions of A/D conversion modes and principals relevant to programming
- Programming examples for the Sequential and Random operating modes

4.2 BASE ADDRESSING

The XVME-566 I/O registers are designed to be addressed within the VMEbus[a] defined 64K short I/O address space, or the upper 64K of the standard memory space. The Data RAM, however, resides only in the standard address space (on any 64K word boundary). Each module connected to the bus must have its own unique address. Thus, the XVME-I/O modules base-addressing scheme has been designed to be switch[a] selectable. When the XVME-566 is installed in the system, it will occupy a 1K byte block of address space (called the I/O Interface Block). The base-addressing scheme for the module is such that the starting address for each I/O Interface Block resides on a 1K boundary. The module base address will, therefore, be one of the 64 1K boundaries available within the short I/O or the upper 64K of the standard memory space. See Table 2-3, Chapter 2 for a complete list of the 64 1K boundaries. The logical registers used for the conversion data on the XVME-566 are given specific addresses within the 1K I/O interface block occupied by the module. These addresses are offset from the module base address. Figure 4-1 shows a representative I/O interface block for the XVME-566 module.

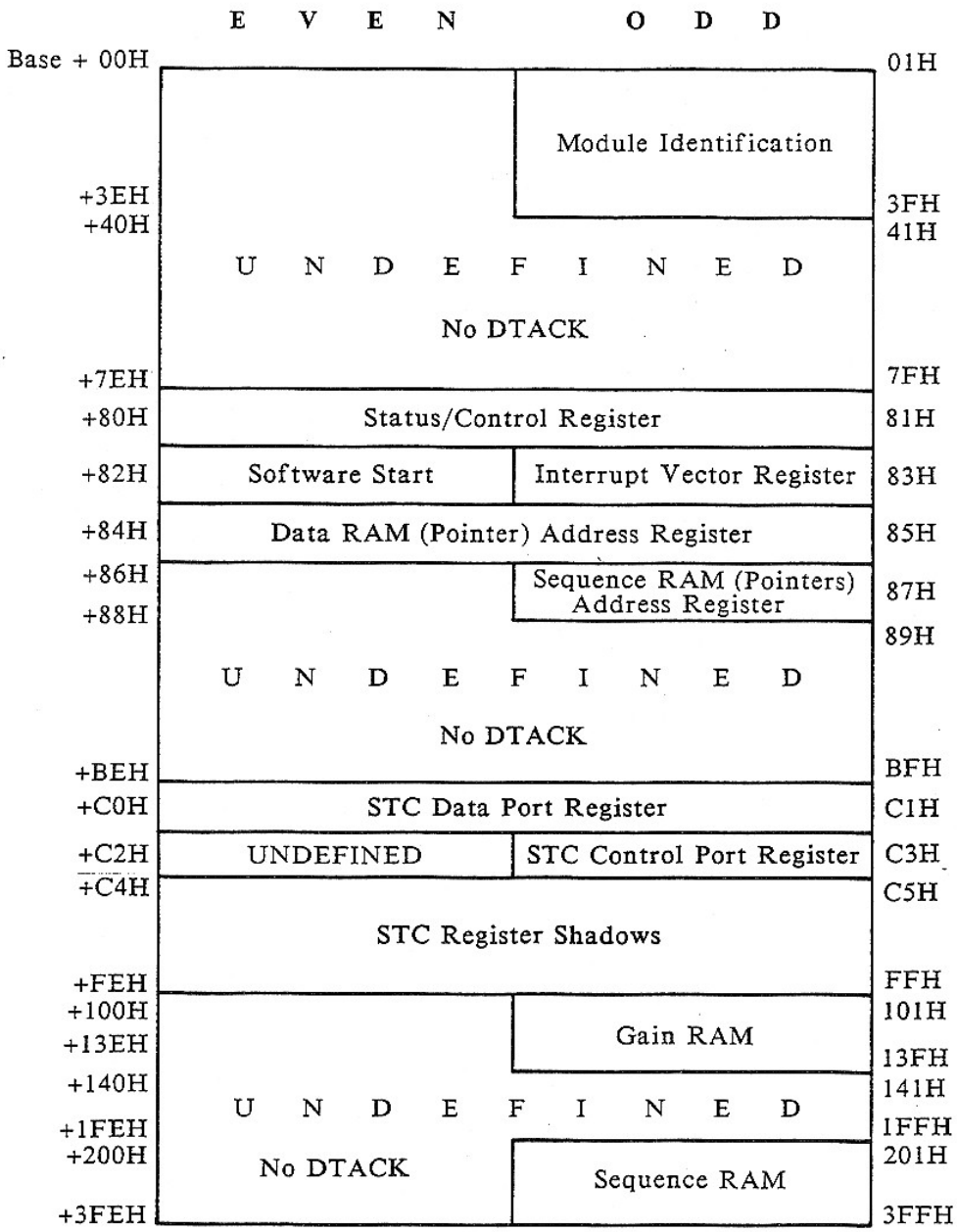


Figure 4-1
The XVME-566 Fast Analog Input Module I/O Interface Block

A specific register on the module can be accessed by adding the specific register offset to the module address. For example, the module's Sequence RAM begins at address 201H within the I/O interface block. Thus, if the module base address is switched to 1000H, the Sequence RAM would begin at address 1201H.

(Module base address)		(Register offset)		(Sequence RAM)
1000H	+	201H	=	1201H

If the XVME-566 is associated with a memory-mapped CPU module (such as the XVME-600 68000 CPU), the short I/O address will be confined to a specific address where it can be accessed by the CPU. If the short I/O address space of the CPU started at FF0000H, for example, then the actual address for the XVME-566 would be FF1000H.

4.3 I/O INTERFACE BLOCK

Each of the following programming locations, of the XVME-566 I/O interface block (previously shown in Figure 4-1), are discussed in greater detail in this chapter's remaining sections:

Module Identification (I.D.PROM) (base + 01H to base + 03FH): (Section 4.3.1) This section includes information concerning the locations specifying model number, manufacturer, and revision levels of the module.

Status/Control Register (base + 80H): (Section 4.3.2) The status/control register contains sixteen single bit locations which provide control signals to reset the module; enable or reset interrupts; selects the mode of analog input operation (random channel or sequential channel); selects mode of data storage (image or contiguous); and controls the status of the red and green LEDs.

Software Start (base + 82H): (Section 4.4.2.2) This write-only register starts data acquisition.

Interrupt Acknowledge (IACK*) Vector Register (base + 83H): (Section 4.3.3) This register holds the vector to be driven onto the VMEbus when an interrupt generated by the input module is acknowledged.

Data RAM (Pointer) Address Register (base + 84H for the high byte; 85H for the low byte): (Section 4.3.5) This 16-bit register holds the pointers to the Data RAM that select the starting addresses that converted data will be stored at within the RAM.

Sequence RAM (Pointer) Address Register (base + 87H): (Section 4.3.6) This 8-bit register steps the sequence controller through the Sequence RAM.

System Timing Controller (base + C0H and base + C3H): Not discussed in this chapter; See STC manual included with the module. The STC Data Port Register is a word address (C0H). The STC Control Port Register is a byte address (C3H).

Gain RAM (base + 101H through base + 13FH, odd bytes): (Section 4.3.7) These memory locations contain the 32-element gain RAM. It must be programmed as part of an input initialization procedure. Specific gains for chosen channels are stored within this 32-byte register.

Sequence RAM (base + 201H through base + 3FFH, odd bytes): (Section 4.3.8) These memory locations control the sequencing of specific channels.

4.3.1 Module Identification (I.D. PROM) (Base + 01H)

The Xycom module identification information for the XVME-566 is located in the odd bytes at addresses 01H to 3FH (base address + 01H to 03FH). The I.D. data is provided as 32 ASCII encoded characters consisting of board type, manufacturer identification, module model number, number of 1K-byte blocks occupied by the module, and module functional revision level. This information can be read by the system processor on power-up to verify the system configuration and operational status. Table 4-1 defines the identification information locations.

Table 4-1
Module I.D. Data

Offset Relative to Module Base	**Contents**	**ASCII Encoding (in hex)**	**Descriptions**
1	V	56	ID PROM identifier, always "VMEID" (5 characters)
3	M	4D	
5	E	45	
7	I	49	
9	D	44	
B	X	58	Manufacturer's I.D., always "XYC" for XYCOM Modules (3 characters)
D	Y	59	
F	C	43	
11	5	35	Module Model Number (3 characters and 4 trailing blanks)
13	6	36	
15	6	36	
17		20	
19		20	
1B		20	
1D		20	
1F	1	31	Number of 1K byte blocks of I/O space occupied by this module (1 character)
21		20	Major functional revision level with leading blank (if single digit)
23	1	31	
25	0	31	Minor functional revision level with trailing blank (if single digit)
27		20	
29	Undefined	00	Manufacturer-dependent information, reserved for future use
2B	Undefined	00	
2D	Undefined	00	
2F	Undefined	00	
31	Undefined	00	
33	Undefined	00	
35	Undefined	00	
37	Undefined	00	
39	Undefined	00	
3B	Undefined	00	
3D	Undefined	00	
3F	Undefined	00	

Each module I.D. data location is accessed by odd VME addresses only. The 32-bytes of ASCII data are assigned to the first 32 odd I/O interface block bytes allowing I.D. information to be accessed by addressing the module base, offset by the specific address for the characters needed. For example, if the base address of the board is jumpered to 1000H, and access is sought for the model number, you will individually add the offset addresses to the base addresses to read the hex-encoded ASCII value at each location.

Table 4-2
Accessing Module I.D. Information

Contents Sought	**Base Address**	**+**	**Offset Address**	**=**	**Code Needed For Access**
5	1000H	+	11H	=	1011H
6	1000H	+	13H	=	1013H
6	1000H	+	15H	=	1015H
20	1000H	+	17H	=	1017H
20	1000H	+	19H	=	1019H
20	1000H	+	1BH	=	101BH
20	1000H	+	1DH	=	101DH

4.3.2 Status/Control Register (Base + word 80H)

The status/control register provides the control signals required to reset the module, enable interrupts, start conversions, selection of operating modes, selection of data storage modes and trigger clock selection. The two operating modes are: sequential and random. The two data storage modes are: contiguous and image. Writing to the status/control register can: enable the sequence controller, select an operating or data storage mode, reset the module, enable module interrupts to the VMEbus, set acquisition accuracy and reset the board. Reading from the status/control register can indicate: whether or not acquisition is in progress (or when acquisition is complete), and if there are interrupts pending.

Figure 4-2 defines the locations in the Status/Control register.

SEE WARNING ON NEXT PAGE

WARNING
Do not use "BSET" or "BCLR" instructions
because separate bit definitions exist in
Status/Control register for read/write
operations.

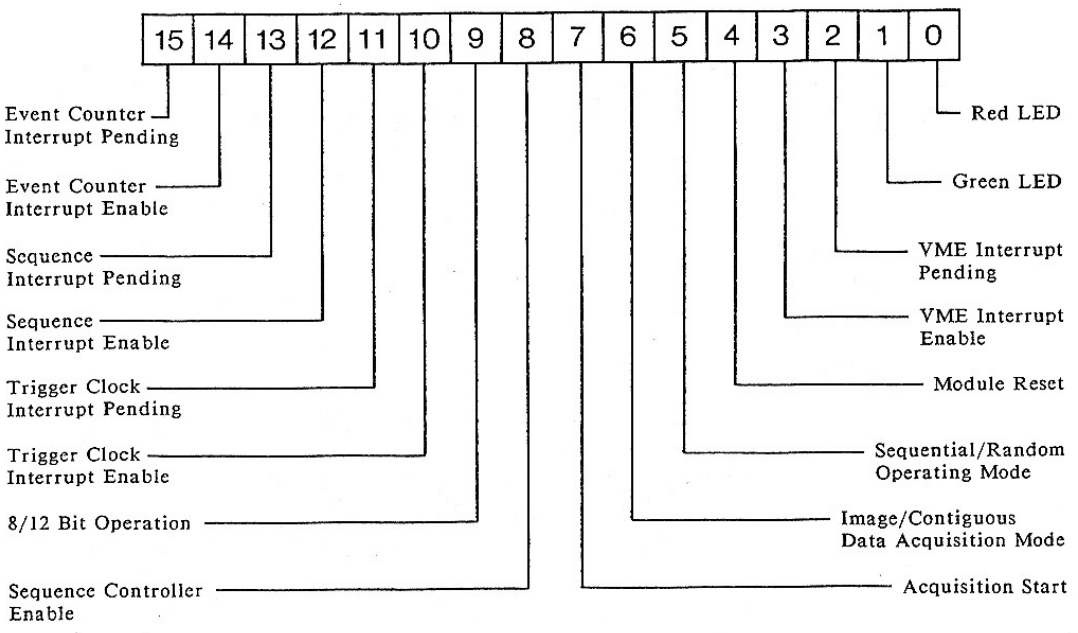


Figure 4-2

Status/Control Register

4.3.2.1 Status Control Register Bit Definitions

The following is a bit-by-bit description of the status/control register:

The Most Significant Byte:

Bit Number	Description
D15	This bit is the event counter "interrupt-pending" flag (STC output 5). A logic `1'READ at this location says an event (interval) has occurred. A logic `1' WRITE clears the interrupt. Logic`0' means "no interrupt pending". An event can be any accumulation of any of the following occurrences: 1. external trigger 2. sample clock 3. trigger clock 4. EOS bit command (from Sequence RAM) 5. STC outputs
D14	A logic `1' written to this location will enable the event counter interrupt.
D13	This is the sequence "interrupt-pending" flag (from bit 5 of the Sequence RAM). A logic `1' READ indicates the occurrence of a sequence interrupt. When a logic `1' WRITE is used, the sequence interrupt will be cleared.
D12	A logic `1' written to this location will enable the sequence interrupt.
D11	This is the trigger clock "interrupt-pending" flag (STC output 2). A logic `1' READ indicates the occurrence of a trigger clock interrupt. When a logic `1' WRITE is used, the trigger clock interrupt will be cleared.
D10	A logic `1' written to this location will enable the trigger clock interrupt.
D9	This bit is assigned the task of choosing 8-bit or 12-bit resolution for conversions. The module will automatically assume a logic `0' (12-bit conversions) upon power-up. A logic `1' WRITE will cause 8-bit conversions.
D8	This is the Sequence Controller enable bit. A logic `1' WRITE to this bit will enable the sequence controller. A logic `0' WRITE will reset the sequence controller. This bit must be set to logic `1' before any data acquisition can be performed.

The Least Significant Byte:

Bit Number	**Descriptions**
D7	READ only. Logic `1' in this location indicates the start of data acquisition.
D6	This bit chooses whether the module will be in either the contiguous mode, or the image mode. Logic `1' (Image Mode) written to this location will cause the Data RAM counter to be loaded with the initial value when bit 7 of the Sequence RAM (the end-of-sequence bit) is also set to logic `1'. The module automatically powers-up at logic `0' (Contiguous Mode). In this mode, the Data RAM pointer will always increment at the end of a conversion after the data is stored.
D5	This bit places the module in either the random or sequential modes. A logic `1' WRITE selects the random mode. A logic `0' selects the sequential mode.
D4	This bit provides a means for a module software reset. If "toggled" to logic `1', then back to logic `0', the sequence controller will reset.
D3	A logic `1' WRITE to this location enables the module to generate VMEbus interrupts.
D2	READ only. This bit is an "interrupt-pending" flag for the VMEbus. A logic `1' READ indicates that one of the three enabled interrupt sources has been set.
D1	This bit controls the green LED. the LED provides visual reference of the module's status. A logic`0' at this site means the green LED is OFF. A logic `1' seen here indicates the green LED is ON.
D0	This bit controls the red LED (and VMEbus SYSFAIL* signal). A logic `0' at this site means the red LED is ON (and SYSFAIL* is driven low). A logic `1' means the red LED is OFF (and SYSFAIL* is driven high).

4.3.3 Interrupt Acknowledge (IACK*) Vector Register (Base + 83H)

The XVME-566 is capable of generating a VMEbus interrupt on any one of the seven levels allowed by VMEbus specification. The Interrupt Acknowledge Vector Register is a READ/WRITE register holding the vector which is automatically driven onto the VMEbus when the module-generated interrupt is acknowledged. This register is accessible at "module base address + 83H".

The interrupt-level switches and IACK* daisy-chain enable jumpers (refer to Table 2.1) are configured when the module is installed. Interrupts are then enabled by writing a logic `1' to bit 3 of the status/control register and one of the three interrupt source enables (see preceding section 4.3.2).

Interrupts are not automatically reset during the VMEbus interrupt acknowledge cycle. Module interrupts must instead be reset by writing a logic `1' to the status/control register bit 15 (event counter), bit 13 (sequence interrupt) or bit 11 (trigger clock). These resets should occur during the interrupt service routine. Pending interrupts may also be reset by a hardware or software reset.

4.3.4 The RAMs

Once initiation has ocurred, the XVME-566 operation is coordinated by a sequence controller that accesses the Gain, Data and Sequence RAMs, requiring trigger and sample clock inputs. Programming for the RAMs is discussed in the sections that follow.

4.3.4.1 Gain RAM (Base + 101H)

The XVME-566 uses a 32-element on-board Gain RAM to store a gain factor for each analog input channel. One of three gain ranges is selected via jumper option at the time the module is installed (See Section 2.7.5).

The Gain RAM is programmed via odd bytes beginning with address Base + 101H and ending with base + 13FH. If the module is operating in the Random Mode, a write to this RAM location may be used to force an A/D conversion.

For convenience, the gain ranges and factors are repeated below.

Input Gain Ranges and Factors

Gain Range	Gain Factors Covered
Range 1 (X1)	1,2,5,10
Range 2 (X4)	4,8,20,40
Range 3 (X10)	10,20,50,100

Initialization

Immediately after power-up or system-reset, the Gain RAM should be programmed (initialized) to provide each input channel (32 SE or 16 DI) with an associated gain factor. Once an input channel is initialized, the associated gain factor will automatically be applied when any A/D conversion occurs on that channel.

Associated with the Sequence RAM, the Gain RAM occupies a 32 x 8 block of memory. Only the two least significant bits are used to select the gain.

Each analog input channel has 1 byte in the Gain RAM associated with it. The Gain RAM byte number corresponds directly with the channel number as indicated in Figure 4-3. When the module is configured for Random channel operation, a conversion can be forced by writing to the Gain RAM byte that corresponds to the channel to be converted.

Figure 4-3 shows how the Gain RAM is arranged.

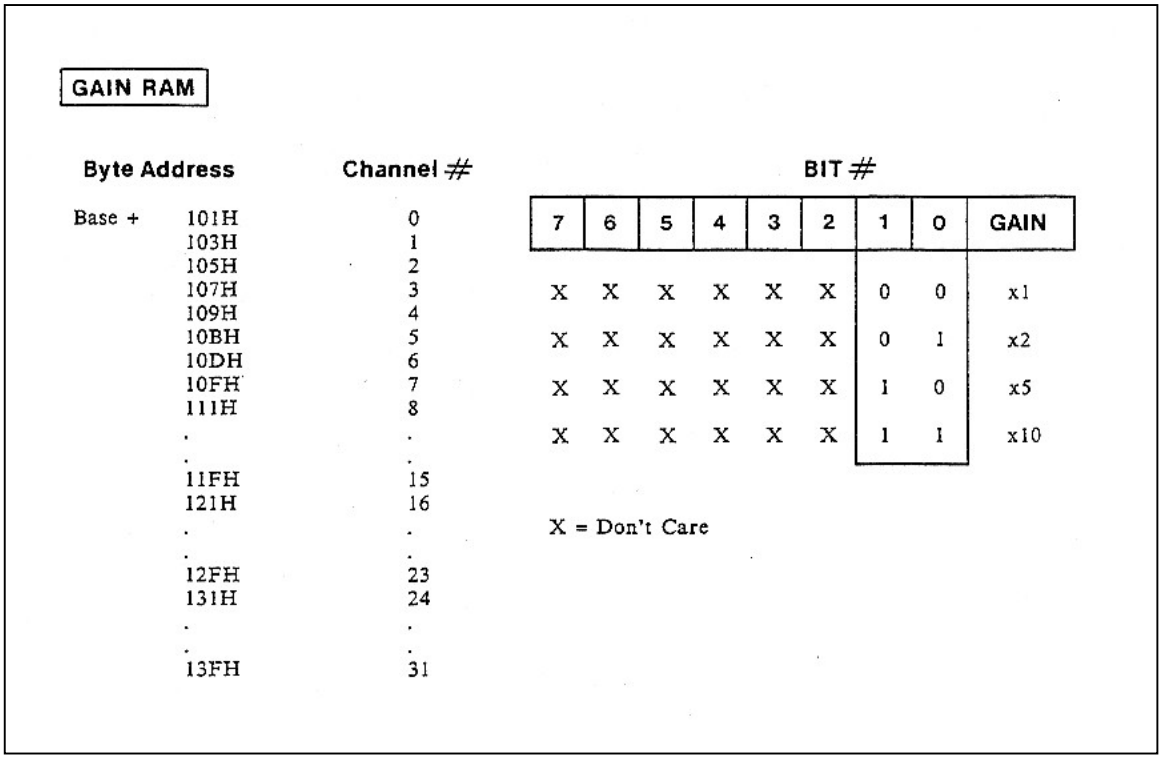


Figure 4-3

Gain RAM

4.3.4.2 Data RAM

The Data RAM is used only for the storage of converted data. There are two possible storage modes: image and contiguous.

Image mode allows the storage of data in a particular area of RAM. This area of RAM is continually updated for a selected number of input channels.

Contiguous mode data is stored from a selected start address. This allows multiple storage of each channel (a history of events on each channel). In both cases channel address is not stored in Data RAM.

4.3.4.3 Data RAM (Pointer) Address Register (Base + word location 84H)

The A/D converter produces digital data which corresponds to the applied analog input from a specified channel. This data is accessible to the `Host' processor via the 32K x 16 Data RAM. To transfer data into the Data RAM, a pointer is needed to track the memory locations. This pointer allows the latching of the starting Data RAM address. The pointer is updated on each conversion (after data has been written). The initial value (starting address) cannot be read after a conversion. However, it is used to reload the pointer in Image Mode when the Sequence RAM EOS bit has been enabled by the sequence controller.

Digital information in this register may be read in either a byte or word format. Figure 4-4 shows how the Data RAM and Data RAM pointer are arranged.

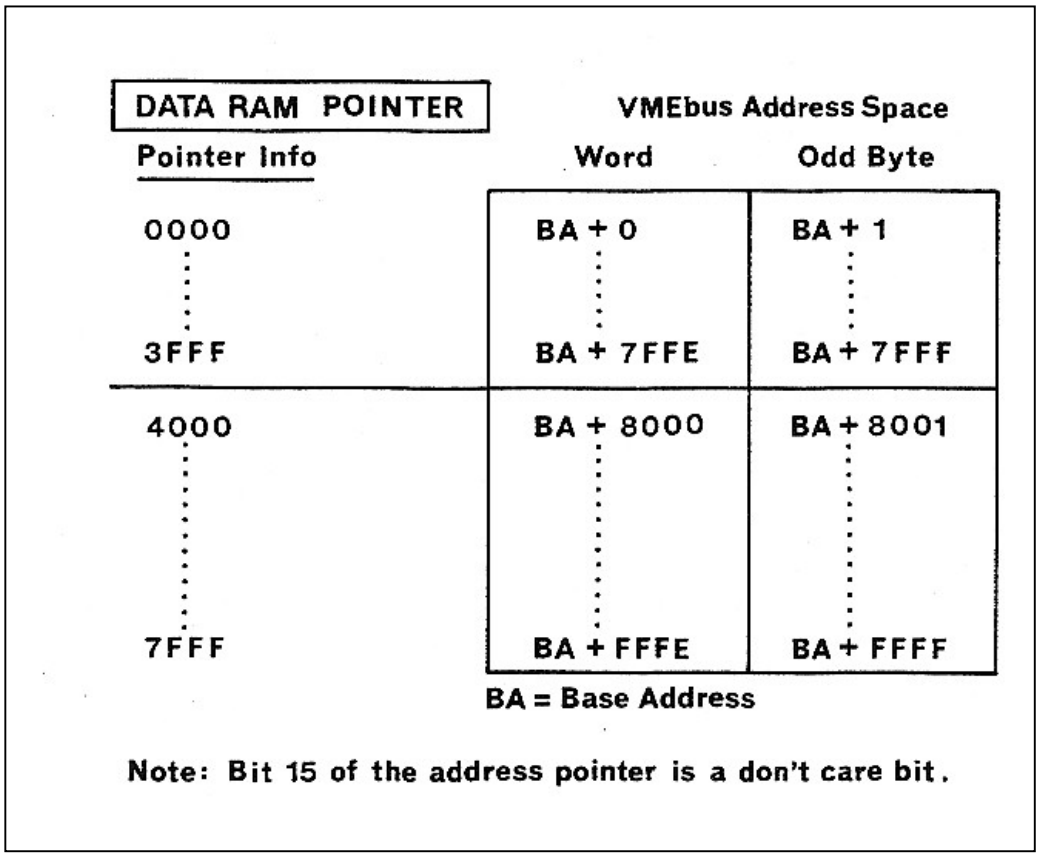


Figure 4-4

Data RAM/Data RAM Pointer

Initialization

Before using the Data RAM, the starting address must be configured. Before data acquisition begins, the Data RAM pointer must be set up to indicate where in the Data RAM the data storage will begin. The simple rule for this is that the Data RAM pointer address is half the Data RAM address. To begin storing data at address 8000 of the Data RAM, the initialized byte in the Data RAM pointer would be 4000.

After data acquisition starts, the Data RAM Pointer will contain the number of the next Data RAM word that will be written to. The Data RAM (or Data RAM pointer) can be read at any time during data acquisition.

4.3.4.4 Sequence RAM (Base + 201H)

The Sequence RAM occupies 256 odd bytes of the I/O address space, starting at base + 201H. All eight bits of this RAM are used to manipulate data. The five least significant bits of this register are used to store input channel addresses. The remaining three bits are control bits. Before data acquisition begins, the Sequence RAM should be initialized with the desired sequence. Figure 4-5 shows the structure of the Sequence RAM byte.

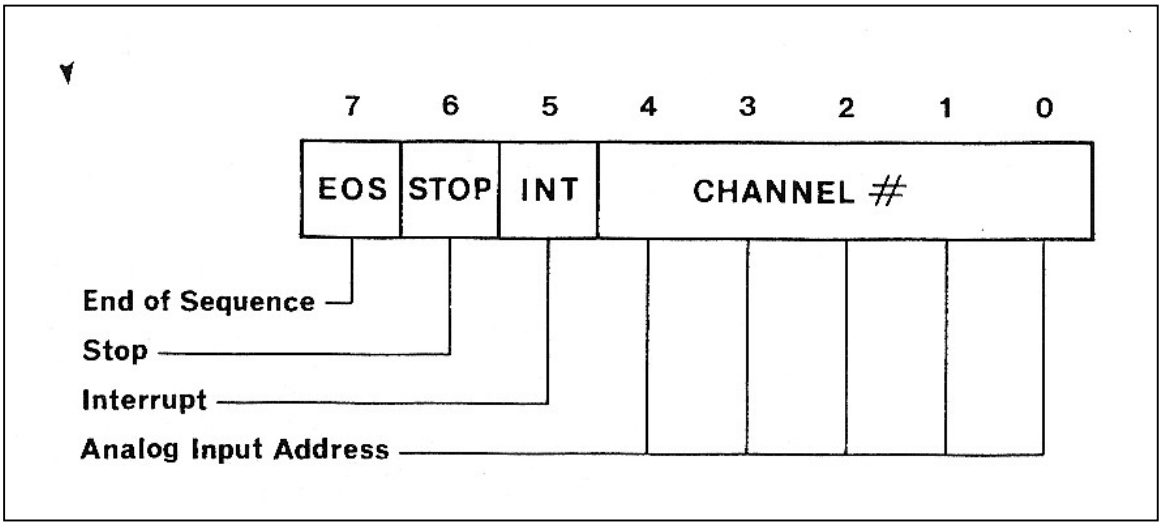


Figure 4-5

Sequence RAM

The following is a bit-by-bit description of the sequence RAM:

Bit Number	**Descriptions**
D7	This is the "end-of-sequence" bit. When set to logic `1', this bit causes the sequence RAM pointer to be set to its initial loaded value. The Data RAM pointer will also be set to its initial value if bit 6 of the status/control register is set for the image mode.
D6	This is the "stop" bit. When set to a logic `1', this bit will halt all activity. Sampling will not begin again until one of the three triggers is received: an external trigger, software trigger or trigger clock. This bit is used with the Sequential Mode only.

D5 — This is the "interrupt" bit. When set to logic`1', this bit works in conjunction with bits 3 and 12 of the status/control register (also set at logic `1') to cause interrupts on the VMEbus (after conversion on its associated channel). After the interrupt is received, it must be cleared with a logic `1' written to bit 13 of the status/control register. This bit can be used for both the Sequential and Random Modes.

D4 – D0 — These are "multiplexer address" bits. The SE and PDI modes use all five bits to address one of 32 channels. The DI mode has only 16 channels, therefore only the four least significant bits are needed for addressing. The Sequence RAM must be loaded with its channel information during initialization. Once data acquisition has begun, information within the RAM cannot be changed.

Table 4-3 shows the bit settings which correspond to the channel addresses.

Table 4-3

SE/DI/PDI Bit Settings
For Channel Addresses

Channel	**Bit Setting**				
Number	**Bit-4**	**Bit-3**	**Bit-2**	**Bit-1**	**Bit-0**
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1
20	1	0	1	0	0
21	1	0	1	0	1
22	1	0	1	1	0
23	1	0	1	1	1
24	1	1	0	0	0
25	1	1	0	0	1
26	1	1	0	1	0
27	1	1	0	1	1
28	1	1	1	0	0
29	1	1	1	0	1
30	1	1	1	1	0
31	1	1	1	1	1

4.3.4.5 Sequence RAM (Pointer) Address Register (Base + 85H)

The Sequence RAM pointer is located in an odd-byte register starting at base address + 85H. At initialization, the Sequence RAM pointer address `00H' corresponds with the Sequence RAM address `201H'. The last pointer address, `FFH', corresponds with the last Sequence RAM address, `3FFH'.

4.4 A/D CONVERSION MODES

The XVME-566 may operate in either of two modes:

MODE

Sequential Channels are converted in sequence beginning with a specified channel (channels may be selected in any numerical order); this mode is most ideal for taking advantage of the module's sampling speed

Random A single A/D conversion is performed on a selected channel

Several conditions may cause a conversion to be initiated. Some of those are:

Sequential Mode
1. Software Trigger
2. Trigger Clock
3. External Trigger

Random Mode
1. Writing a byte to the Gain RAM or Sequence RAM

4.4.1 Sequential Mode Programming

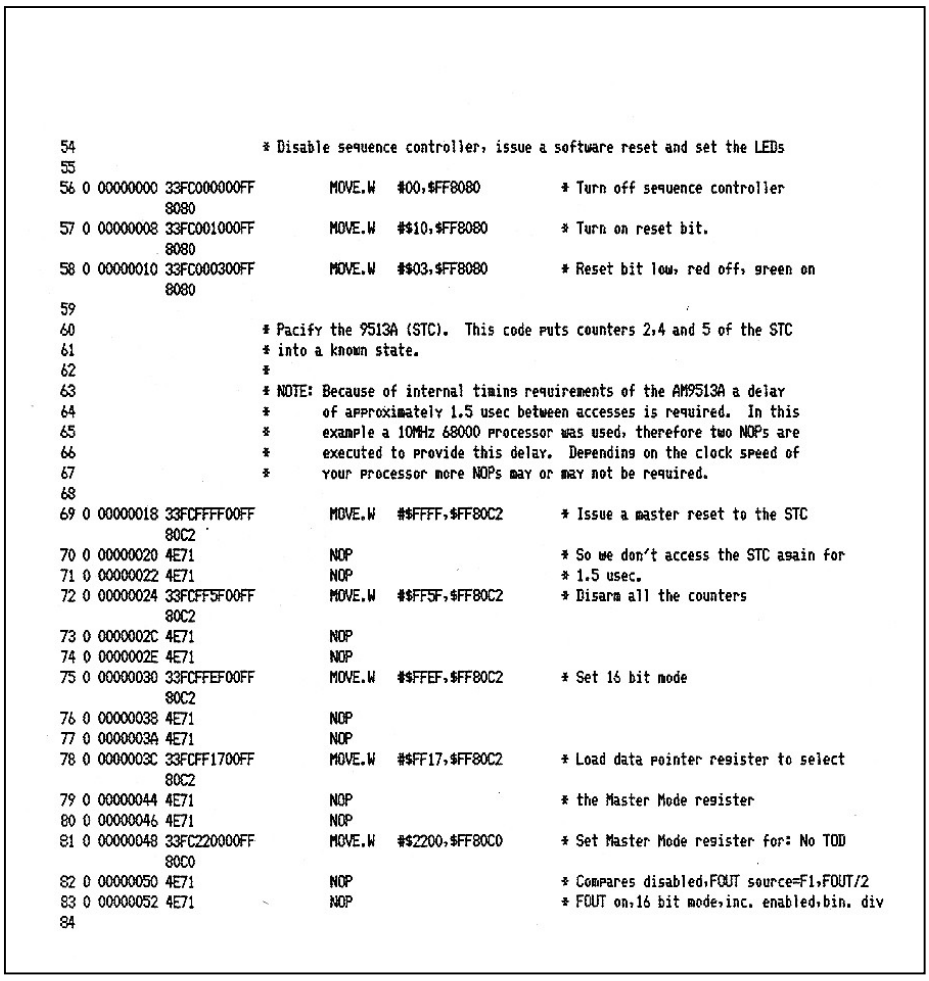
The Sequential Mode converts data at high speeds on one to 32 channels. The mode is invoked by setting bit 5 of the status/control register to a logic `0'. The module powers-up in the sequential mode.

The following example (Figure 4-6) shows the initialization of the XVME-566 for continuous conversions on channels 31-0 with Data RAM in Image Mode. The event counter and trigger clocks are not used.

```
1                ************************************************************
2                *                                                          *
3                *        Example Sequential Mode initialization code       *
4                *        for the XVME-566. The followins are the steps     *
5                *        required for proper initialization of the         *
6                *        XVME-566:                                         *
7                *                                                          *
8                *        1. Disable the Sequence Controller, this can be   *
9                *           done by reseting bit 8 of the Status/Control   *
10               *           register.                                      *
11               *                                                          *
12               *        2. Issue a module reset.                          *
13               *                                                          *
14               *        3. Pacify counters 2, 4 and 5 (Trigger clock,     *
15               *           Sample Clock and Event Counter) of the STC.    *
16               *           This step will put these counters into a       *
17               *           known state.                                   *
18               *                                                          *
19               *        4. Initialize STC counter 4 (Sample Clock) as a   *
20               *           100KHz clock.                                  *
21               *                                                          *
22               *        5. Initialize any other counters that you wish    *
23               *           use for your application, ie. Trigger Clock    *
24               *           or Event Counter                               *
25               *                                                          *
26               *        6. Initialize Gain RAM                            *
27               *                                                          *
28               *        7. Write the desired channel scaning sequence     *
29               *           into the sequence RAM                          *
30               *                                                          *
31               *        8. Initialize the Data RAM and Sequence RAM       *
32               *           pointers with their starting addresses         *
33               *                                                          *
34               *        9. Enable the sequence controller                 *
35               *                                                          *
36               *       10. Start data aquisition by providing a trigger   *
37               *                                                          *
38               *                                                          *
39               *        NOTE: This example assumes a module bass address  *
40               *              offset in the Short I/O of 8000H and a      *
41               *              Data RAM base address of 900000H. This      *
42               *              example also does not use the Trigger       *
43               *              Clock or the Event Counter therefore they   *
44               *              are not initialized.                        *
45               *                                                          *
46               ************************************************************
47
48               * WARNING: Before attempting to initialize the XVME-566 be sure
49               *          the sequence controller is disabled.  The sequence controller ca
50               *          be disabled by clearing bit 8 in the Status/Control register
51               *          ( Base address + 80H).
52
```

Figure 4-6

Initializing Sequential Mode



```
54                      * Disable sequence controller, issue a software reset and set the LEDs
55
56 0 00000000 33FC000000FF          MOVE.W   #00,$FF8080          * Turn off sequence controller
             8080
57 0 00000008 33FC001000FF          MOVE.W   #$10,$FF8080         * Turn on reset bit.
             8080
58 0 00000010 33FC000300FF          MOVE.W   #$03,$FF8080         * Reset bit low, red off, green on
             8080
59
60                      * Pacify the 9513A (STC).  This code puts counters 2,4 and 5 of the STC
61                      * into a known state.
62                      *
63                      * NOTE: Because of internal timing requirements of the AM9513A a delay
64                      *       of approximately 1.5 usec between accesses is required.  In this
65                      *       example a 10MHz 68000 processor was used, therefore two NOPs are
66                      *       executed to provide this delay.  Depending on the clock speed of
67                      *       your processor more NOPs may or may not be required.
68
69 0 00000018 33FCFFFF00FF          MOVE.W   #$FFFF,$FF80C2       * Issue a master reset to the STC
             80C2
70 0 00000020 4E71                  NOP                           * So we don't access the STC again for
71 0 00000022 4E71                  NOP                           * 1.5 usec.
72 0 00000024 33FCFF5F00FF          MOVE.W   #$FF5F,$FF80C2       * Disarm all the counters
             80C2
73 0 0000002C 4E71                  NOP
74 0 0000002E 4E71                  NOP
75 0 00000030 33FCFFEF00FF          MOVE.W   #$FFEF,$FF80C2       * Set 16 bit mode
             80C2
76 0 00000038 4E71                  NOP
77 0 0000003A 4E71                  NOP
78 0 0000003C 33FCFF1700FF          MOVE.W   #$FF17,$FF80C2       * Load data pointer register to select
             80C2
79 0 00000044 4E71                  NOP                           * the Master Mode register
80 0 00000046 4E71                  NOP
81 0 00000048 33FC220000FF          MOVE.W   #$2200,$FF80C0       * Set Master Mode register for: No TOD
             80C0
82 0 00000050 4E71                  NOP                           * Compares disabled,FOUT source=F1,FOUT/2
83 0 00000052 4E71                  NOP                           * FOUT on,16 bit mode,inc. enabled,bin. div
84
```

Figure 4-6

Initializing the Sequential Mode (cont'd)

```
85                          * Disable Counter 2 (Trigger Clock), TC output pulled high
86
87 0 00000054 33FCFF0200FF          MOVE.W   #$FF02,$FF80C2       * Select counter 2 mode register
              80C2
88 0 0000005C 4E71                  NOP
89 0 0000005E 4E71                  NOP
90 0 00000060 33FC0B0200FF          MOVE.W   #$0B02,$FF80C0       * Set counter 2 for TC toggle mode
              80C0
91 0 00000068 4E71                  NOP
92 0 0000006A 4E71                  NOP
93 0 0000006C 33FCFFEA00FF          MOVE.W   #$FFEA,$FF80C2       * Set counter 2 TC output high
              80C2
94 0 00000074 4E71                  NOP
95 0 00000076 4E71                  NOP
96
98                          * Disable Counter 4 (Sample Clock), TC output pulled high.
99
100 0 00000078 33FCFF0400FF         MOVE.W   #$FF04,$FF80C2       * Select counter 4 mode register
              80C2
101 0 00000080 4E71                 NOP
102 0 00000082 4E71                 NOP
103 0 00000084 33FC0B0200FF         MOVE.W   #$0B02,$FF80C0       * Set counter 4 for TC toggle mode
              80C0
104 0 0000008C 4E71                 NOP
105 0 0000008E 4E71                 NOP
106 0 00000090 33FCFFEC00FF         MOVE.W   #$FFEC,$FF80C2       * Set counter TC output high
              80C2
107 0 00000098 4E71                 NOP
108 0 0000009A 4E71                 NOP
109
110                         * Disable Counter 5 (Event Counter), TC output pulled high.
111
112 0 0000009C 33FCFF0500FF         MOVE.W   #$FF05,$FF80C2       * Select counter 5 mode register
              80C2
113 0 000000A4 4E71                 NOP
114 0 000000A6 4E71                 NOP
115 0 000000A8 33FC0B0200FF         MOVE.W   #$0B02,$FF80C0       * Set counter 5 for TC toggle mode
              80C0
116 0 000000B0 4E71                 NOP
117 0 000000B2 4E71                 NOP
118 0 000000B4 33FCFFED00FF         MOVE.W   #$FFED,$FF80C2       * Set counter 5 TC output high
              80C2
119 0 000000BC 4E71                 NOP
120 0 000000BE 4E71                 NOP
121
```

Figure 4-6

Initializing the Sequential Mode (cont'd)

```
122                         * Initialize counter 4 (sample clock) as a 100KHz rate generator
123
124 0 000000C0 33FCFF0400FF         MOVE.W   #$FF04,$FF80C2        * Select counter 4 mode register
               80C2
125 0 000000C8 4E71                 NOP
126 0 000000CA 4E71                 NOP
127 0 000000CC 33FC95A500FF         MOVE.W   #$95A5,$FF80C0        * Set counter 4 as sample clock: Active
               80C0
128 0 000000D4 4E71                 NOP                            * low TC,Mode 0,downcount,count source:S5
129 0 000000D6 4E71                 NOP                            * (FOUT),falling edge,gate:G4,active high
130 0 000000D8 33FC001400FF         MOVE.W   #$0014,$FF80C0        * Counter 4 downcount value
               80C0
131 0 000000E0 4E71                 NOP
132 0 000000E2 4E71                 NOP
133 0 000000E4 33FCFF6800FF         MOVE.W   #$FF68,$FF80C2        * Load and arm counter 4
               80C2
134
135                         * NOTE: If any other counters are going to used (ie. Trigger Clock, Event
136                         *       Counter.), they should be initialized at this time.
137
139                         * Initialize Gain RAM for all 32 channels to a gain of 1.
140
141 0 000000EC 303C001F               MOVE.W   #32-1,D0              * Adjust loop counter for DBRA instr.
142 0 000000F0 41F900FF8101          LEA.L    $FF8101,A0            * Address of gain RAM start
143 0 000000F6               SETGAIN
144 0 000000F6 10BC0000              MOVE.B   #0,(A0)               * Set this channel for gain of 1
145 0 000000FA 5488                  ADDQ.L   #2,A0                 * Bump the address to next location
146 0 000000FC 51C8FFF8              DBRA     D0,SETGAIN            * Continue until all 32 are written
147
148                         * Initialize the Sequence RAM for the desired sequence.  This example
149                         * will use a sequence of 31 - 0 to illustrate the boards flexibility.
150
151 0 00000100 41F900FF8201          LEA.L    $FF8201,A0            * Start of Sequence RAM
152 0 00000106 303C001F              MOVE.W   #32-1,D0              * Initial channel # & loop counter
153 0 0000010A               SEQLOOP
154 0 0000010A 1080                  MOVE.B   D0,(A0)               * Set sequence RAM
155 0 0000010C 5488                  ADDQ.L   #2,A0                 * Bump address pointer
156 0 0000010E 51C8FFFA              DBRA     D0,SEQLOOP
157
158                         * For this example we will set the Data RAM for the image mode and allow the
159                         * board to do repeated conversions on all 32 channels.  In order to do this
160                         * we must set the EOS bit (bit  #7) in the last byte of our sequence or channel
161                         * 0.
162
163 0 00000112 103900FF823F            MOVE.B   $FF8201+62,D0         * Get the byte that contains channel 0
164 0 00000118 00000080              OR.B     #%10000000,D0         * Set bit 7 165 0 0000011C 13C000FF823F          MOVE.B   D0,$F
F8201+62        * Write new byte to sequence RAM
166
```

Figure 4-6

Initializing the Sequential Mode (cont'd)

```
167                         * Load Data RAM and Sequence RAM pointers with their initial values
168
169 0 00000122 33FC000000FF          MOVE.W   #0,$FF8084          * Start at beginning of Data RAM
               8084
170 0 0000012A 13FC000000FF          MOVE.B   #0,$FF8087          * Start at beginning of Sequence RAM
               8087
171
172                         * Enable the Sequence controller, reset any pending interrupts and set the
173                         * data RAM in image mode.
174
175 0 00000132 33FCA9430FF           MOVE.W   #$A943,$FF8080      * Initialize Status/Control register
               8080
176
177                         * Begin the sequence by writing to the software trigger location (FF8082).
178
179 0 0000013A 13FC000000FF          MOVE.B   #0,$FF8082          * Issue software trigger
               8082
180
181 0 00000142 4E4F                  TRAP     #15
182 0 00000144 0000                  DC.W     0
183
184
185
186
187
188
189
190
191                                  END

****** TOTAL ERRORS      0--

****** TOTAL WARNINGS    0--
```

Figure 4-6

Initializing the Sequential Mode (cont'd)

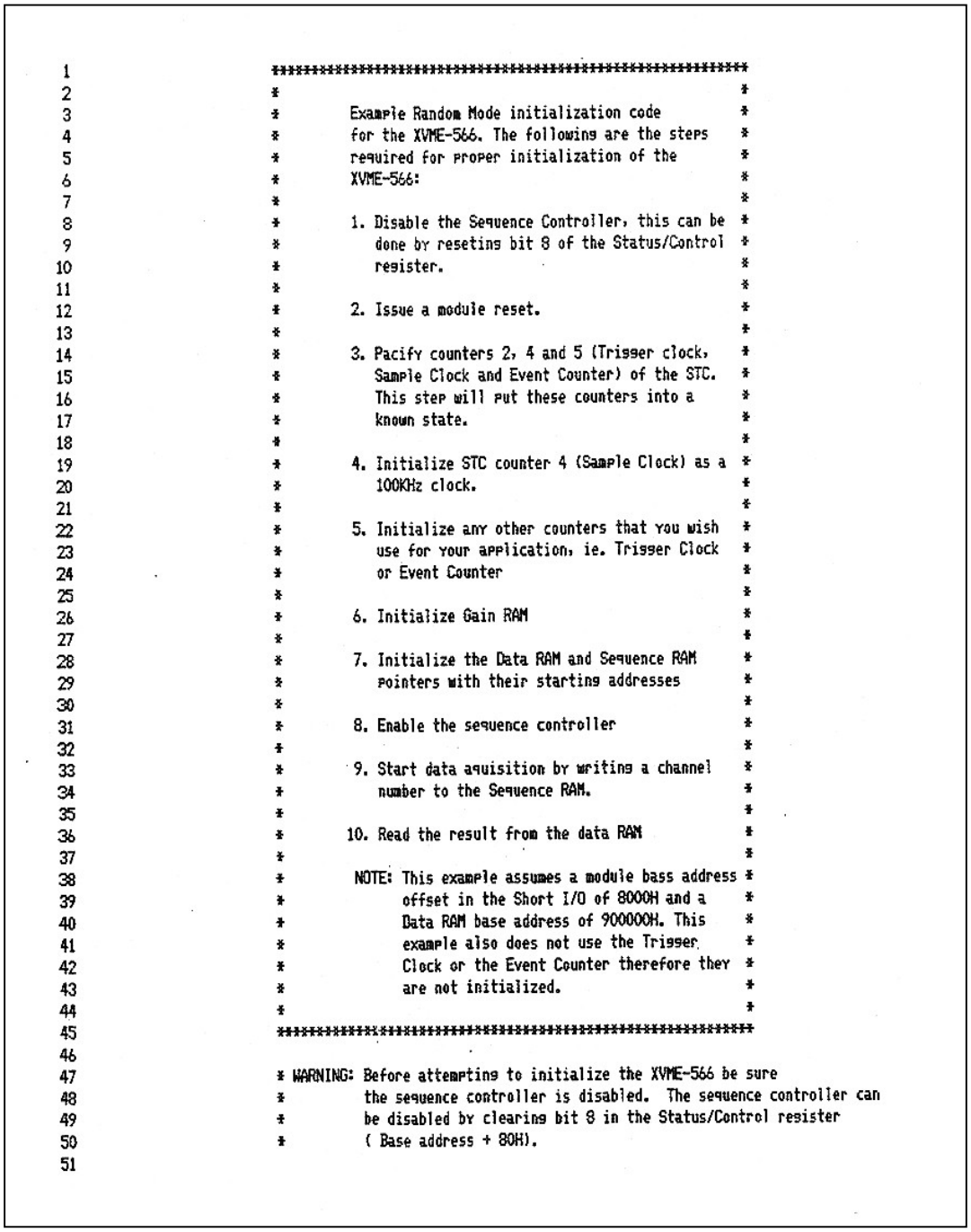
NOTE

When communicating with the STC, all codes must be in hex. In addition, all codes written for the 16-bit data bus to the command port (of the STC) should include the prefix `FF', as illustrated in Table 4-4. It should also be noted that all accesses to the STC should be word accesses.

4.4.2 Random Mode Selection

The Random Mode converts data one channel at a time. It is invoked by writing a logic `1' to status/control register bit 5. Once in the mode, a write to the Gain and Sequence RAMs (with channel number and control bits) will start the acquisition and conversion on the specified channel.

The following pages discuss the programming of the XVME-566 (Random Mode) in more detail. Figure 4-7 is a programming example showing sample versions of each of the steps listed in this section.



```
1   ********************************************************
2   *                                                      *
3   *       Example Random Mode initialization code        *
4   *       for the XVME-566. The following are the steps  *
5   *       required for proper initialization of the      *
6   *       XVME-566:                                      *
7   *                                                      *
8   *       1. Disable the Sequence Controller, this can be *
9   *          done by reseting bit 8 of the Status/Control *
10  *          register.                                   *
11  *                                                      *
12  *       2. Issue a module reset.                       *
13  *                                                      *
14  *       3. Pacify counters 2, 4 and 5 (Trigger clock,  *
15  *          Sample Clock and Event Counter) of the STC. *
16  *          This step will put these counters into a    *
17  *          known state.                                *
18  *                                                      *
19  *       4. Initialize STC counter 4 (Sample Clock) as a *
20  *          100KHz clock.                               *
21  *                                                      *
22  *       5. Initialize any other counters that you wish *
23  *          use for your application, ie. Trigger Clock *
24  *          or Event Counter                            *
25  *                                                      *
26  *       6. Initialize Gain RAM                         *
27  *                                                      *
28  *       7. Initialize the Data RAM and Sequence RAM    *
29  *          pointers with their starting addresses      *
30  *                                                      *
31  *       8. Enable the sequence controller              *
32  *                                                      *
33  *       9. Start data aquisition by writing a channel  *
34  *          number to the Sequence RAM.                 *
35  *                                                      *
36  *      10. Read the result from the data RAM           *
37  *                                                      *
38  *       NOTE: This example assumes a module bass address *
39  *             offset in the Short I/O of 8000H and a   *
40  *             Data RAM base address of 900000H. This   *
41  *             example also does not use the Trigger    *
42  *             Clock or the Event Counter therefore they *
43  *             are not initialized.                     *
44  *                                                      *
45  ********************************************************
46
47  * WARNING: Before attempting to initialize the XVME-566 be sure
48  *          the sequence controller is disabled.  The sequence controller can
49  *          be disabled by clearing bit 8 in the Status/Control register
50  *          ( Base address + 80H).
51
```

Figure 4-7

Initializing Random Mode

```
53                          * Disable sequence controller, issue a software reset and set the LEDs
54
55 0 00000000 33FC000000FF          MOVE.W   #00,$FF8080          * Turn off sequence controller
             8080
56 0 00000008 33FC001000FF          MOVE.W   #$10,$FF8080         * Turn on reset bit.
             8080
57 0 00000010 33FC000300FF          MOVE.W   #$03,$FF8080         * Reset bit low, red off, green on
             8080
58
59                          * Pacify the 9513A (STC).  This code puts counters 2,4 and 5 of the STC
60                          * into a known state.
61                          *
62                          * NOTE: Because of internal timing requirements of the AM9513A a delay
63                          *       of approximately 1.5 usec between accesses is required.  In this
64                          *       example a 10MHz 68000 processor was used, therefore two NOPs are
65                          *       executed to provide this delay.  Depending on the clock speed of
66                          *       your processor more NOPs may or may not be required.
67
68 0 00000018 33FCFFFF00FF          MOVE.W   #$FFFF,$FF80C2       * Issue a master reset to the STC
             80C2
69 0 00000020 4E71                  NOP                           * So we don't access the STC again for
70 0 00000022 4E71                  NOP                           * 1.5 usec.
71 0 00000024 33FCFF5F00FF          MOVE.W   #$FF5F,$FF80C2       * Disarm all the counters
             80C2
72 0 0000002C 4E71                  NOP
73 0 0000002E 4E71                  NOP
74 0 00000030 33FCFFEF00FF          MOVE.W   #$FFEF,$FF80C2       * Set 16 bit mode
             80C2
75 0 00000038 4E71                  NOP
76 0 0000003A 4E71                  NOP
77 0 0000003C 33FCFF1700FF          MOVE.W   #$FF17,$FF80C2       * Load data pointer register to select
             80C2
78 0 00000044 4E71                  NOP                           * the Master Mode register
79 0 00000046 4E71                  NOP
80 0 00000048 33FC2200000FF         MOVE.W   #$2200,$FF80C0       * Set Master Mode register for: No TOD
             80C0
81 0 00000050 4E71                  NOP                           * Compares disabled,FOUT source=F1,FOUT/2
82 0 00000052 4E71                  NOP                           * FOUT on,16 bit mode,inc. enabled,bin. div
83
84                          * Disable Counter 2 (Trigger Clock), TC output pulled high
85
86 0 00000054 33FCFF0200FF          MOVE.W   #$FF02,$FF80C2       * Select counter 2 mode register
             80C2
87 0 0000005C 4E71                  NOP
88 0 0000005E 4E71                  NOP
89 0 00000060 33FC0B0200FF          MOVE.W   #$0B02,$FF80C0       * Set counter 2 for TC toggle mode
             80C0
90 0 00000068 4E71                  NOP
91 0 0000006A 4E71                  NOP
92 0 0000006C 33FCFFEA00FF          MOVE.W   #$FFEA,$FF80C2       * Set counter 2 TC output high
             80C2
93 0 00000074 4E71                  NOP
94 0 00000076 4E71                  NOP
95
```

Figure 4-7

Initializing Random Mode (cont'd)

```
 97                         * Disable Counter 4 (Sample Clock), TC output pulled high.
 98
 99 0 00000078 33FCFF0400FF         MOVE.W   #$FF04,$FF80C2       * Select counter 4 mode register
               80C2
100 0 00000080 4E71                 NOP
101 0 00000082 4E71                 NOP
102 0 00000084 33FC0B0200FF         MOVE.W   #$0B02,$FF80C0       * Set counter 4 for TC toggle mode
               80C0
103 0 0000008C 4E71                 NOP
104 0 0000008E 4E71                 NOP
105 0 00000090 33FCFFEC00FF         MOVE.W   #$FFEC,$FF80C2       * Set counter TC output high
               80C2
106 0 00000098 4E71                 NOP
107 0 0000009A 4E71                 NOP
108
109                         * Disable Counter 5 (Event Counter), TC output pulled high.
110
111 0 0000009C 33FCFF0500FF         MOVE.W   #$FF05,$FF80C2       * Select counter 5 mode register
               80C2
112 0 000000A4 4E71                 NOP
113 0 000000A6 4E71                 NOP
114 0 000000A8 33FC0B0200FF         MOVE.W   #$0B02,$FF80C0       * Set counter 5 for TC toggle mode
               80C0
115 0 000000B0 4E71                 NOP
116 0 000000B2 4E71                 NOP
117 0 000000B4 33FCFFED00FF         MOVE.W   #$FFED,$FF80C2       * Set counter 5 TC output high
               80C2
118 0 000000BC 4E71                 NOP
119 0 000000BE 4E71                 NOP
120
121                         * Initialize counter 4 (sample clock) as a 100KHz rate generator
122
123 0 000000C0 33FCFF0400FF         MOVE.W   #$FF04,$FF80C2       * Select counter 4 mode register
               80C2
124 0 000000C8 4E71                 NOP
125 0 000000CA 4E71                 NOP
126 0 000000CC 33FC95A500FF         MOVE.W   #$95A5,$FF80C0       * Set counter 4 as sample clock: Active
               80C0
127 0 000000D4 4E71                 NOP                           * low TC,Mode 0,downcount,count source:S5
128 0 000000D6 4E71                 NOP                           * (FOUT),falling edge,gate:G4,active high
129 0 000000D8 33FC00140FF          MOVE.W   #$0014,$FF80C0       * Counter 4 downcount value
               80C0
130 0 000000E0 4E71                 NOP
131 0 000000E2 4E71                 NOP
132 0 000000E4 33FCFF6800FF         MOVE.W   #$FF68,$FF80C2       * Load and arm counter 4
               80C2
133
134                         * NOTE: If any other counters are going to used (ie. Trigger Clock, Event
135                         *       Counter.), they should be initialized at this time.
136
```

Figure 4-7

Initializing Random Mode (cont'd)

```
138                         * Initialize Gain RAM for all 32 channels to a gain of 1.
139
140 0 000000EC 303C001F             MOVE.W   #32-1,D0            * Adjust loop counter for DBRA instr.
141 0 000000F0 41F900FF8101         LEA.L    $FF8101,A0          * Address of gain RAM start
142 0 000000F6              SETGAIN
143 0 000000F6 10BC0000             MOVE.B   #0,(A0)             * Set this channel for gain of 1
144 0 000000FA 5488                 ADDQ.L   #2,A0               * Bump the address to next location
145 0 000000FC 51C8FFF8             DBRA     D0,SETGAIN          * Continue until all 32 are written
146
147                         * For this example we will set the Data RAM for the image mode and allow all
148                         * the conversions to be stored in the first word of the data RAM.  This will
149                         * be done by settting the Data RAM Pointer to 0 and setting the EOS bit when we
150                         * write the number of the channel to be converted to the sequence RAM, this
151                         * will cause the Data RAM Pointer to be reloaded after the conversion is done.
152                         * We will also set the interrupt bit so thatwe will have a flag the will tell
153                         * us when the conversion has been completed and the results have been stored
154                         * in the Data RAM.
155
156                         * Load Data RAM and Sequence RAM pointers with their initial values.  For
157                         * Random Mode the contents of the Sequence RAM should have no affect on
158                         * the conversion, but we will set it to 0 just in case.
159
160 0 00000100 33FC000000FF            MOVE.W   #0,$FF8084          * Start at beginning of Data RAM
                8084
161 0 00000108 13FC000000FF            MOVE.B   #0,$FF8087          * Set Sequence RAM Pointer to 0
                8087
162
163                         * Enable the Sequence controller, reset any pending interrupts, set for
164                         * Random Mode conversions and put data RAM in image mode.
165
166 0 00000110 33FCA96300FF          MOVE.W   #$A963,$FF8080      * Initialize Status/Control register
                8080
167
168                         * Here we will start a conversion on channel 0 by writing an $A0 to the
169                         * first byte of the Sequence RAM, this will tell the module to convert
170                         * channel 0, reload the Data RAM Pointer with it's orginal contents and
171                         * set the Sequence Interrupt pending bit in the Status/Control register
172                         * when the conversion is complete.
173
174 0 00000118 13FC00A000FF            MOVE.B   #$A0,$FF8201        * Force a conversion on channel 0
                8201
175
176                         * Now wait for the sequence interrupt bit in the Status/Control register to
177                         * go active, this will tell us when the results of the conversion have been
178                         * stored in Data RAM.
179                         *
180                         * NOTE: The Sequence Interrupt pending bit will have to be reset before the
181                         * next conversion. This should be done by storing in a data register the bit
182                         * pattern to be written to the Status/Control register to reset the interupt.
183                         * Then write the contents of the data register into the Status/Control register
184                         * This should be done because of the different read/write defintions of some of
185                         * the bits in the Status/Control register. A bit type operation like BCLR
186                         * perform a read of the Status/Control register first then set the indicated
187                         * bit and write the data back to the Status/Control register if any other
188                         * interrupt pending bits were set they would now be reset resulting in that
189                         * interrupt being missed by the system.
190
191
```

Figure 4-7

Initializing Random Mode (cont'd)

```
192
193 0 00000120             WAIT
194 0 00000120 103900FF8080         MOVE.B   $FF8080,D0          * Get contents of Status/Control register
195 0 00000126 08000005             BTST.L   #5,D0               * Test Seq. Int. pending bit
196 0 0000012A 67F4                 BEQ.S    WAIT                * Wait if bit not set
197 0 0000012C 103C0021             MOVE.B   #%00100001,D0       * Reset Seq. Int. pending bit and leave
198 0 00000130 13C000FF8080         MOVE.B   D0,$FF8080          * Sequence Contoller enabled
199 0 00000136 323900900000         MOVE.W   $900000,D1          * Read the results from Data RAM.
200
201 0 0000013C 4E4F                 TRAP     #15                 * Stop and return to monitor
202 0 0000013E 0000                 DC.W     0
203
204                                 END

****** TOTAL ERRORS      0--
****** TOTAL WARNINGS    0--
```

Figure 4-7

Initializing Random Mode (cont'd)

The VMEbus is prevented from accessing the module for 625nSec both when the Sample Clock becomes active and at the end of a conversion (so data can be stored in the Data RAM). Figure 4-8 shows the timing sequence involved for Random Mode operation.

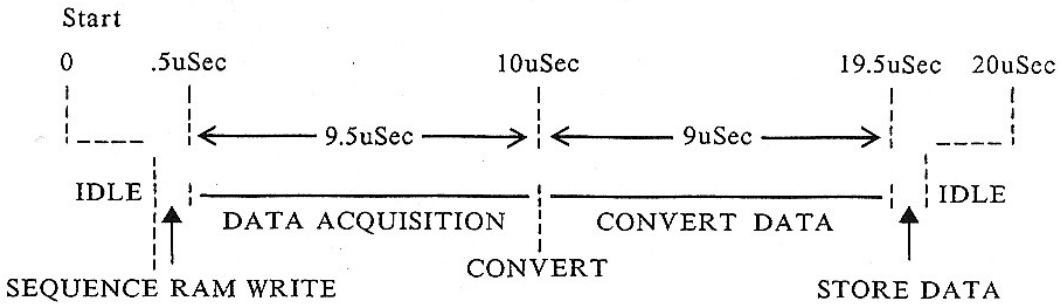


Figure 4-8

Random Mode Time Sequence

4.4.3 Triggered Conversions

The XVME-566 uses several types of triggers to start conversions. The type of trigger is determined by the particular need. The three types of triggers discussed in the following sections are: Trigger Clock, Software Trigger and External Trigger.

4.4.3.1 Trigger Clock

The Trigger Clock is used to start or re-start a sampling sequence. Working in conjunction with the sample clock, the trigger clock determines when the board is ready to begin conversions, then starts the conversion process.

If the Trigger Clock is to be used to periodically re-start a sequence, the routine simulated in Figure 4-8 must be run during STC initialization at the start of the program. The purpose of the routine is to assure that the minimum period for the Trigger Clock is greater than the length of the sequence controlled by the Sample Clock. The time line illustrated in Figure 4-9 shows the relationship between the Trigger Clock and the Sample Clock.

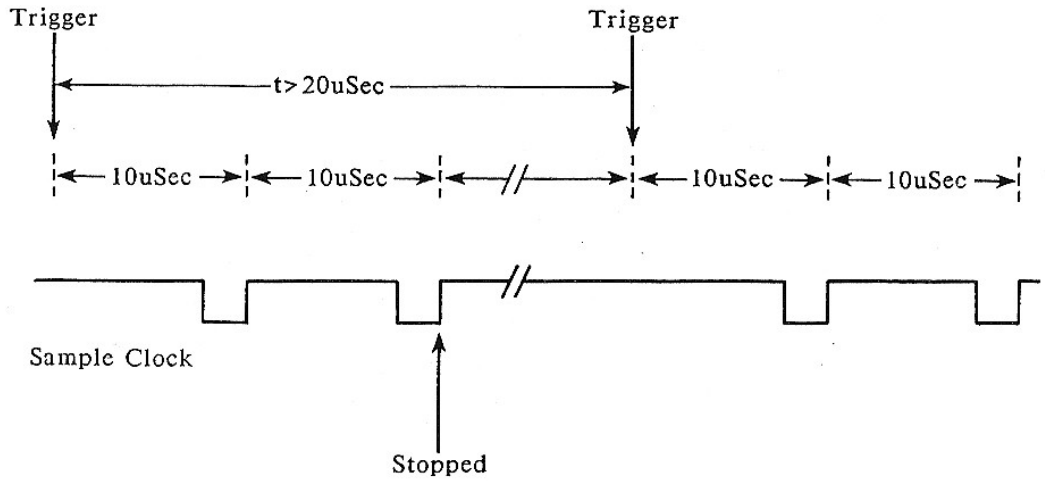


Figure 4-9

Initializing for the Trigger Clock

Initialization

If the Trigger Clock output of the STC is used to re-start the sequence, the Trigger Clock must be initialized (in addition to the Sample Clock) at the beginning of the program. On the next few pages, Figure 4-10 illustrates the use of the Trigger Clock in Sequential Mode.

```
1                ************************************************************
2                *                                                          *
3                *        Example Sequential Mode with Trigger Clock        *
4                *        initialization code for the XVME-566              *
5                *        The following are the steps required for proper   *
6                *        initialization of the XVME-566 using the Trigger  *
7                *        Clock:                                            *
8                *                                                          *
9                *        1. Disable the Sequence Controller, this can be   *
10               *           done by reseting bit 8 of the Status/Control   *
11               *           register.                                      *
12               *                                                          *
13               *        2. Issue a module reset.                          *
14               *                                                          *
15               *        3. Pacify counters 2, 4 and 5 (Trigger clock,     *
16               *           Sample Clock and Event Counter) of the STC.    *
17               *           This step will put these counters into a       *
18               *           known state.                                   *
19               *                                                          *
20               *        4. Initialize STC counter 4 (Sample Clock) as a   *
21               *           100KHz clock.                                  *
22               *                                                          *
23               *        5. Initialize counter 2 as a 2KHz Trigger clock   *
24               *                                                          *
25               *        6. Initialize any other counters your specifc     *
26               *           application may require.                       *
27               *                                                          *
28               *        7. Initialize Gain RAM                            *
29               *                                                          *
30               *        8. Write the desired channel scaning sequence     *
31               *           into the sequence RAM                          *
32               *                                                          *
33               *        9. Initialize the Data RAM and Sequence RAM       *
34               *           pointers with their starting addresses         *
35               *                                                          *
36               *       10. Enable the sequence controller                 *
37               *                                                          *
38               *       11. Start data aquisition by providing a trigger   *
39               *                                                          *
40               *                                                          *
41               *        NOTE: This example assumes a module bass address  *
42               *              offset in the Short I/O of 8000H and a      *
43               *              Data RAM base address of 900000H. This      *
44               *              example also initializes the Trigger        *
45               *              Clock as a 2KHz rate generator which will   *
46               *              trigger a sequence once every 500usec.      *
47               *                                                          *
48               ************************************************************
49
50               * WARNING: Before attempting to initialize the XVME-566 be sure
51               *          the sequence controller is disabled.  The sequence controller can
52               *          be disabled by clearing bit 8 in the Status/Control register
53               *          ( Base address + 80H).
54
```

Figure 4-10

Initializing the Trigger Clock in Sequential Mode

```
56                          * Disable sequence controller, issue a software reset and set the LEDs
57
58 0 00000000 33FC000000FF          MOVE.W   #00,$FF8080          * Turn off sequence controller
             8080
59 0 00000008 33FC001000FF          MOVE.W   #$10,$FF8080         * Turn on reset bit.
             8080
60 0 00000010 33FC000300FF          MOVE.W   #$03,$FF8080         * Reset bit low, red off, green on
             8080
61
62                          * Pacify the 9513A (STC).  This code puts counters 2,4 and 5 of the STC
63                          * into a known state.
64                          *
65                          * NOTE: Because of internal timing requirements of the AM9513A a delay
66                          *       of approximately 1.5 usec between accesses is required.  In this
67                          *       example a 10MHz 68000 processor was used, therefore two NOPs are
68                          *       executed to provide this delay.  Depending on the clock speed of
69                          *       your processor more NOPs may or may not be required.
70
71 0 00000018 33FCFFFF00FF          MOVE.W   #$FFFF,$FF80C2       * Issue a master reset to the STC
             80C2
72 0 00000020 4E71                  NOP                           * So we don't access the STC again for
73 0 00000022 4E71                  NOP                           * 1.5 usec.
74 0 00000024 33FCFF5F00FF          MOVE.W   #$FF5F,$FF80C2       * Disarm all the counters
             80C2
75 0 0000002C 4E71                  NOP
76 0 0000002E 4E71                  NOP
77 0 00000030 33FCFFEF00FF          MOVE.W   #$FFEF,$FF80C2       * Set 16 bit mode
             80C2
78 0 00000038 4E71                  NOP
79 0 0000003A 4E71                  NOP
80 0 0000003C 33FCFF1700FF          MOVE.W   #$FF17,$FF80C2       * Load data pointer register to select
             80C2
81 0 00000044 4E71                  NOP                           * the Master Mode register
82 0 00000046 4E71                  NOP
83 0 00000048 33FC220000FF          MOVE.W   #$2200,$FF80C0       * Set Master Mode register for: No TOD
             80C0
84 0 00000050 4E71                  NOP                           * Compares disabled,FOUT source=F1,FOUT/2
85 0 00000052 4E71                  NOP                           * FOUT on,16 bit mode,inc. enabled,bin. div
86
87                          * Disable Counter 2 (Trigger Clock), TC output pulled high
88
89 0 00000054 33FCFF0200FF          MOVE.W   #$FF02,$FF80C2       * Select counter 2 mode register
             80C2
90 0 0000005C 4E71                  NOP
91 0 0000005E 4E71                  NOP
92 0 00000060 33FC0B0200FF          MOVE.W   #$0B02,$FF80C0       * Set counter 2 for TC toggle mode
             80C0
93 0 00000068 4E71                  NOP
94 0 0000006A 4E71                  NOP
95 0 0000006C 33FCFFEA00FF          MOVE.W   #$FFEA,$FF80C2       * Set counter 2 TC output high
             80C2
96 0 00000074 4E71                  NOP
97 0 00000076 4E71                  NOP
98
```

Figure 4-10

Initializing the Trigger Clock in Sequential Mode (cont'd)

Figure 4-10

Initializing the Trigger Clock in Sequential Mode (cont'd)

```
100                         * Disable Counter 4 (Sample Clock), TC output pulled high.
101
102 0 00000078 33FCFF0400FF         MOVE.W   #$FF04,$FF80C2       * Select counter 4 mode register
               80C2
103 0 00000080 4E71                  NOP
104 0 00000082 4E71                  NOP
105 0 00000084 33FC0B0200FF         MOVE.W   #$0B02,$FF80C0       * Set counter 4 for TC toggle mode
               80C0
106 0 0000008C 4E71                  NOP
107 0 0000008E 4E71                  NOP
108 0 00000090 33FCFFEC00FF         MOVE.W   #$FFEC,$FF80C2       * Set counter TC output high
               80C2
109 0 00000098 4E71                  NOP
110 0 0000009A 4E71                  NOP
111
112                         * Disable Counter 5 (Event Counter), TC output pulled high.
113
114 0 0000009C 33FCFF0500FF         MOVE.W   #$FF05,$FF80C2       * Select counter 5 mode register
               80C2
115 0 000000A4 4E71                  NOP
116 0 000000A6 4E71                  NOP
117 0 000000A8 33FC0B0200FF         MOVE.W   #$0B02,$FF80C0       * Set counter 5 for TC toggle mode
               80C0
118 0 000000B0 4E71                  NOP
119 0 000000B2 4E71                  NOP
120 0 000000B4 33FCFFED00FF         MOVE.W   #$FFED,$FF80C2       * Set counter 5 TC output high
               80C2
121 0 000000BC 4E71                  NOP
122 0 000000BE 4E71                  NOP
123
124                         * Initialize counter 4 (sample clock) as a 100KHz rate generator
125
126 0 000000C0 33FCFF0400FF         MOVE.W   #$FF04,$FF80C2       * Select counter 4 mode register
               80C2
127 0 000000C8 4E71                  NOP
128 0 000000CA 4E71                  NOP
129 0 000000CC 33FC95A500FF         MOVE.W   #$95A5,$FF80C0       * Set counter 4 as sample clock: Active
               80C0
130 0 000000D4 4E71                  NOP                           * low TC,Mode 0,downcount,count source:S5
131 0 000000D6 4E71                  NOP                           * (FOUT),falling edge,gate:G4,active high
132 0 000000D8 33FC001400FF         MOVE.W   #$0014,$FF80C0       * Counter 4 downcount value
               80C0
133 0 000000E0 4E71                  NOP
134 0 000000E2 4E71                  NOP
135 0 000000E4 33FCFF6800FF         MOVE.W   #$FF68,$FF80C2       * Load and arm counter 4
               80C2
136
```

Figure 4-10

Initializing the Trigger Clock in Sequential Mode (cont'd)

```
138                     * Initialize counter 2 (Trigger Clock) as a 2KHz rate generator which will
139                     * trigger a sequence once every 500usec.
140                     *
141                     *CAUTION: The period of the Trigger Clock must be greater than the total time
142                     *         required to convert and store all the channels specified in the
143                     *         sequence. It will take 330 usec to convert and store all 32 channels
144                     *         therfore the peroid of the trigger clock must be greater than
145                     *         330 usec.  For this example we will use a trigger clock period of
146                     *         500 usec.
147
148 0 000000EC 33FCFF0200FF        MOVE.W   #$FF02,$FF80C2       * Select counter 2 mode resister
             80C2
149 0 000000F4 4E71             NOP
150 0 000000F6 4E71             NOP
151 0 000000F8 33FC152500FF     MOVE.W   #$1525,$FF80C0       * Set counter 2 as Trigger Clock: No gate,
             80C0
152 0 00000100 4E71             NOP                           * source=S5(FOUT),falling edge,no spec.gate
153 0 00000102 4E71             NOP                           * load reg., binary, downcount
154 0 00000104 33FC03E800FF     MOVE.W   #1000,$FF80C0        * Counter 2 downcount value
             80C0
155 0 0000010C 4E71             NOP
156 0 0000010E 4E71             NOP
157 0 00000110 33FCFF4200FF     MOVE.W   #$FF42,$FF80C2       * Load counter wait to arm
             80C2
158
160                     * Initialize Gain RAM for all 32 channels to a gain of 1.
161
162 0 00000118 303C001F         MOVE.W   #32-1,D0             * Adjust loop counter for DBRA instr.
163 0 0000011C 41F900FF8101     LEA.L    $FF8101,A0           * Address of gain RAM start
164 0 00000122          SETGAIN
165 0 00000122 10BC0000         MOVE.B   #0,(A0)              * Set this channel for gain of 1
166 0 00000126 5488             ADDQ.L   #2,A0                * Bump the address to next location
167 0 00000128 51C8FFF8         DBRA     D0,SETGAIN           * Continue until all 32 are written
168
169                     * Initialize the Sequence RAM for the desired sequence.  This example
170                     * will use a sequence of 31 - 0 to illustrate the boards flexibility.
171
172 0 0000012C 41F900FF8201       LEA.L    $FF8201,A0           * Start of Sequence RAM
173 0 00000132 303C001F         MOVE.W   #32-1,D0             * Initial channel # & loop counter
174 0 00000136          SEQLOOP
175 0 00000136 1080             MOVE.B   D0,(A0)              * Set sequence RAM
176 0 00000138 5488             ADDQ.L   #2,A0                * Bump address pointer
177 0 0000013A 51C8FFFA         DBRA     D0,SEQLOOP
178
179                     * For this example we will set the Data RAM for the image mode and allow the
180                     * board to do repeated conversions on all 32 channels. We will set bits 7
181                     * (EOS) and 6 (STOP) in the last byte of our sequence.  This will cause the
182                     * Sequence and Data RAM pointers to be reloaded with their initial values
183                     * and stop the sequence when the last channel has been converted.  The Trigger
184                     * Clock will the restart the sequence.
185
```

Figure 4-10

Initializing the Trigger Clock in Sequential Mode (cont'd)

```
186
187 0 0000013E 103900FF823F              MOVE.B      $FF8201+62,D0       * Get the byte that contains channel 0
188 0 00000144 000000C0                 OR.B        #%11000000,D0       * Set bits 6 and 7 (STOP and EOS)
189 0 00000148 13C000FF823F             MOVE.B      D0,$FF8201+62       * Write new byte to sequence RAM
190
191                     * Load Data RAM and Sequence RAM pointers with their initial values
192
193 0 0000014E 33FC000000FF             MOVE.W      #0,$FF8084          * Start at beginning of Data RAM
                8084
194 0 00000156 13FC000000FF             MOVE.B      #0,$FF8087          *Start at beginning of Sequence RAM
                8087
195
196                     * Enable the Sequence controller, reset any pending interrupts and set the
197                     * data RAM in image mode
198
199 0 0000015E 33FCA94300FF             MOVE.W      #$A943,$FF8080      *Initialize Status/Control register
                8080
200
201                     *Begin the sequence by arming STC counter 2 (Trigger Clock)
202
203 0 00000166 33FCFF2200FF             MOVE.W      #$FF22,$FF80C2      * Arm STC counter 2
                80C2
204
205 0 0000016E 4E4F                     TRAP        #15
206 0 00000170 0000                     DC.W        0
207
208
209
210
211
212
213
214
215                                     END


****** TOTAL ERRORS          0--
****** TOTAL WARNINGS        0--
```

Figure 4-10

Initializing the Trigger Clock in Sequential Mode (cont'd)

4.4.3.2 Software Trigger (Base + 82H)

The Software Trigger is the simplest method for starting or re-starting a sequence. It is activated with a WRITE to module address base + 82H as follows:

```
CLR.B   SOFTRIG
```

4.4.3.3. Interrupts

The analog input portion of the module can generate an interrupt to notify the host that the A/D conversion is complete and the results are available. The level and vector generated by this interrupt are both user-selectable.

The following three steps must be performed in order to generate an interrupt:

1. Interrupt level select switches must be configured to enable the module IACK* handling circuitry. (See Section 2.6.6.1).

2. The Interrupt Vector Register (location base + 83H) must be loaded with the required vector. This vector register will be read by the interrupt handler when the interrupt is acknowledged.

3. Interrupts must be enabled via bits 3 and 12 in the status/control register (See Section 4.3.2).

At the completion of a conversion, (if bit 5 of the Sequence RAM is set) an interrupt will be generated. Interrupts can also be generated by an event counter or a trigger clock.

Chapter 5

INPUT CALIBRATION

5.1 INTRODUCTION

Calibration facilities have been provided on the XVME-566 Fast Analog Input Module for the analog input circuits. It is recommended that calibrations be checked any time the module is re-configured (i.e., new inputs added, jumpers changed, etc).

The analog input circuit must be calibrated to assure the accuracy of a 12 bit system. Several points of calibration must be performed. These points include:

-- the Instrumentation Amp
-- the Offset Adjustment at the A/D Converter
-- the Gain Adjustment at the A/D Converter

The calibrations must be performed in the ordered listed above.

5.2 PROGRAMMABLE GAIN OFFSET ADJUSTMENT

The following procedure is recommended in performing the calibration of the instrumentation amplifier. Offset adjustments must be made for the input and output stages of the LH0084 (U79). The input adjustment is made via R20 whereas the output adjustment is made at the A/D converter.

Procedure

1. Set resistor R20 to center position.

2. Insert jumper J12 (for single-ended mode) or jumpers J12 and J11 (for differential mode).

3. Set input multiplexer address to channel zero.

4. Set input Stage Gain to 1 (insert jumper J7A and J7B and program the GAIN Ram for CH0 to 0).

5. Select channel 0.

6. Measure and record the output voltage (V01) at TP3 (TP4 is Ground).

7. Set input stage gain to 10 by loading 03H into CH0 of the GAIN Ram.

8. Cause a selection of CH0.

9. Measure and record the output voltage (V010) at TP3.

10. Calculate the offset voltage contributed by V01 and V010 with the following equation:

    Voos = 1/9 x [(10 x V01) - V010]

11. While maintaining an input stage gain of ten, adjust the input offset voltage potentiometer (R20) until the output at TP3 equals Voos, ± 30 uV.

12. Insert jumper J7 to desired Gain range as depicted in Table 2.15.

13. Remove jumper(s) J12 (or J12 and J11).

5.3 ANALOG-TO-DIGITAL OFFSET and GAIN ADJUSTMENT

The following adjustments must be made to the A/D stages to complete calibration of the module.

Procedure

1. Null the Instrumentation Amp.

2. Set J7 to the desired gain range.

3. Perform continuous conversions on Channel 0.

4. Display the conversion result on a CRT screen (in HEX format) for verification.

5. Adjust the offset potentiometer (R18 for unipolar; R16 for bipolar) until the conversion result is toggling equally between the two points described in Table 5-1. This can be accomplished by applying O volts + 1/2 LSB to Channel 0 (1.22mV for 0-10V, or ± 5V; and 2.44mV for ± 10V).

6. Adjust the gain. This is accomplished by applying the full scale voltage of that particular range minus 1-1/2 LSB, and adjusting R15 until toggling occurs between the points described in Table 5-1. MAKE SURE THAT JUMPERS J11 AND J12 HAVE BEEN REMOVED.

Table 5-1

A/D Calibration Points

Analog Input	0V + 1/2 LSB	Transition Points F.S. - 1-1/2 LSB	Jumpers
Unipolar (Straight Binary)	0000H 0001H	0FFEH 0FFFH	J5,J16A
Bipolar (Offset Binary)	0800H 0801H	0FFEH 0FFFH	J5 Out, J16A
Bipolar (Two's Complement Offset Binary)	0000H 0001H	07FEH 07FFH	J5,J16B

5.4 DEFAULT JUMPER CONFIGURATION

The XVME-566 is shipped in a test-ready state. That is, the module is pre-tested and pre-configured to simplify initial calibration procedures. Table 5-2, below, lists the jumpers installed on the module before delivery.

Table 5-2

XVME-566 Jumpers Installed Before Delivery

Jumper	Description
J1B	Connects INTERRUPT ACKNOWLEDGE IN to INTERRUPT ACKNOWLEDGE OUT from daisy chain
J2B	Enables INTERRUPT ACKNOWLEDGE IN to daisy chain
J3B	Enables module for short I/O map
J5	Analog-to-straight binary conversion
J7A	Differential Amp output Stage Gain of X1
J7B	Differential Amp output Stage Gain of X1
J8A	Voltage range selector to ADC; ± 10V
J9A	Configures module for SE operation; accompanies J9C
J9C	Configures module for SE operation; accompanies J9A
J10A	Unipolar voltage range selector with potentiometer XX
J13A	External trigger selection provision
J14	Grounds JK1 Pin 49 for external trigger reference
J15	Configures module for SE operation; accompaniesJ9Aor J9D
J16B	Zeroes (0) bits 12-15 on converted data

Appendix A

XYCOM STANDARD I/O ARCHITECTURE

INTRODUCTION

This Appendix defines XYCOM's Standard I/O Architecture for XVME I/O modules. This Standard I/O Architecture has been incorporated on all XVME-I/O modules in order to provide a simple and consistent method of programming the entire module line. The I/O Architecture specifies the logical aspects of bus interfaces, as opposed to the "physical" or electrical aspects as defined in the VMEbus specifications. The module elements which are standardized by the Xycom I/O Architecture are listed below:

- **Module Addressing -** Where a module is positioned in the I/O address space and how software can read from or write to it.
- **Module Identification -** How software can identify which modules are installed in a system.
- **Interrupt Control** – How software is able to control and monitor the capability of the module to interrupt the system
- **Module Operational Status -** How the operator can (through software) determine the operational condition of specific modules within the system.
- **Communication Between Modules -** How master (host) processors and intelligent I/O modules communicate through shared global memory or the dual-access RAM on the I/O modules.
- **The I/O Kernel**– How intelligent and non-intelligent "kernels" facilitate the operation of all XYCOM I/O modules

MODULE ADDRESSING

All Xycom I/O modules are designed to be addressed within the VMEbus-defined 64 Kbyte short I/O address space. The restriction of I/O modules to the short I/O address space provides separation of program/data address space and the I/O address space. This convention simplifies software design and minimizes hardware and module cost, while at the same time, providing 64 Kbytes of address space for I/O modules.

Base Addressing

Since each I/O module connected to the bus must have its own unique base address, the base addressing scheme for Xycom VME I/O modules has been designed to be switch-selectable. Each XVME-I/O module installed in the system requires at least a 1 Kbyte block of the short address space. Thus, each I/O module has a base address that starts on a 1 Kbyte boundary. As a result, the Xycom I/O modules have all been implemented to decode base addresses in 1 Kbyte (400H) increments.

Figure A-1 shows an abbreviated view of the short I/O memory.

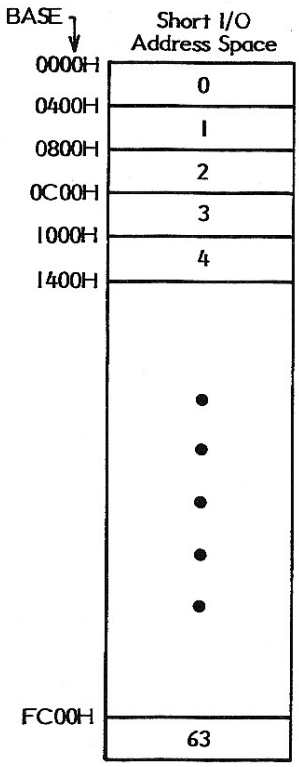


Figure A-1.
64 Kbyte Short I/O Address Space
For Modules Occupying a 1K Block

Standardized Module I/O Map

I/O Interface Block
This 1 Kbyte block of short I/O addresses allocated to each XVME module is mapped with a standardized format to simplify programming and data access. The locations of frequently used registers and module-specific identification information are uniform.

For example, the module identification is always found in the first 32 odd bytes of the module memory block -- with these addresses being relative to the jumpered base address (i.e., module I.D. data address = base address + odd bytes 1H - 3FH).

The byte located at base address +81H on each module contains a status/control register which provides the results of diagnostics for verification of the module's operational condition.

Module-Specific
The next area of the module I/O interface block (base address + 82H - up to FFFH is module-specific and varies in size from one module to the next. It is in this area that the module holds specific I/O status, data, and pointer registers for use with IPC protocol.

All intelligent XVME-I/O modules have an area of their I/O Interface Blocks defined as dual access RAM. This area of memory provides the space where XVME slave I/O modules access their command blocks and where XVME master modules could access their command blocks (i.e., master modules can also access global system memory).

The remainder of the I/O interface block is then allocated to various module-specific tasks, registers, buffers, ports, etc.

Figure A-2 shows an address map of an XVME I/O module interface block, and how it relates to the VMEbus short I/O address space. Notice that any location in the I/O Interface Block may be accessed by simply using the **address formula**:

Module Base Address + Relative Offset = Desired Location

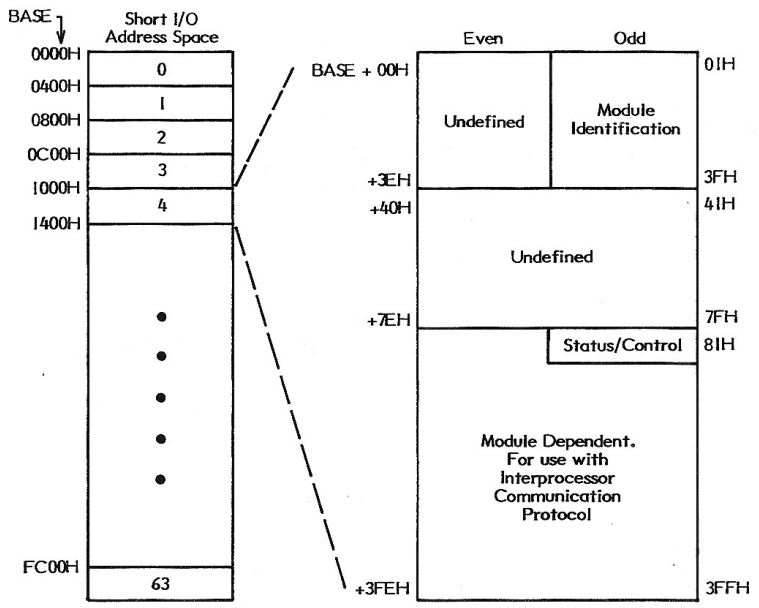


Figure A-2.
XVME-I/O Module Address Map

MODULE SPECIFIC IDENTIFICATION DATA

The module identification scheme provides a unique method of registering module specific information in an ASCII encoded format. The I.D. data is provided as thirty-two ASCII encoded characters consisting of the board type, manufacturer identification, module model number, number of 1 Kbyte blocks occupied by the module, and module functional revision level information. this information can be studied by the system processor on power-up to verify the system configuration and operational status. Table A-1, on the following page, defines the identification information locations.

Table A-1. Module I.D. Data

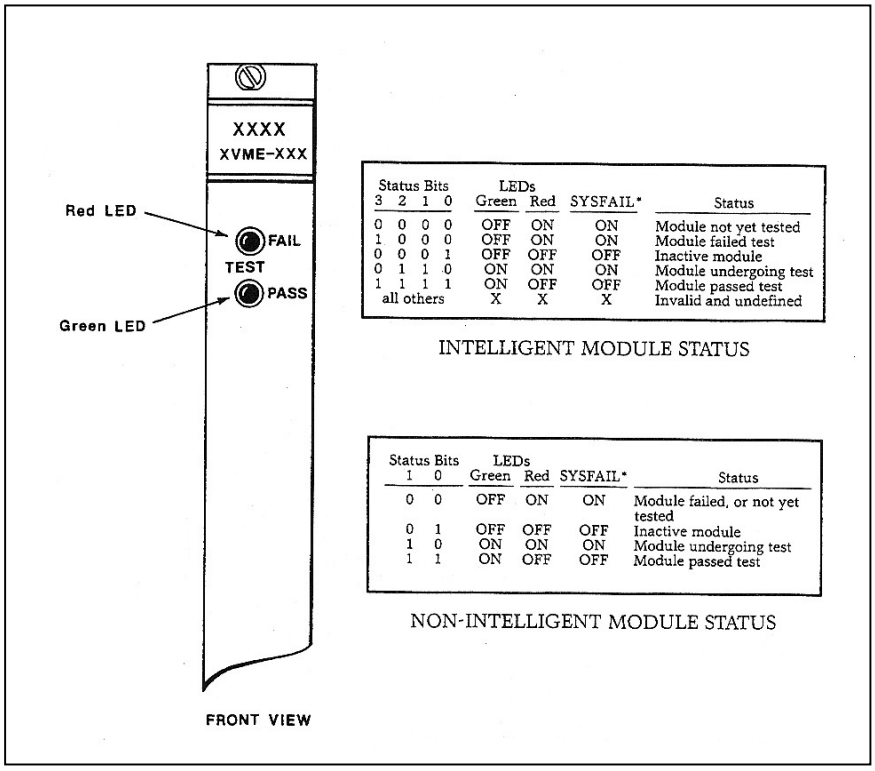
Offset Relative to Module Base	Contents	ASCII Encoding in Hex	Descriptions
1	V	56	ID PROM identifier, always "VMEID" (5 characters)
3	M	4D	
5	E	45	
7	I	49	
9	D	44	
B	X	58	Manufacturer's I.D.
D	Y	59	
F	C	43	Modules (3 characters)
11	5	35	Module model number (4 characters and 3 trailing blanks)
13	6	36	
15	6	36	
17		20	
19		20	
1B		20	
1D		20	
1F	1	31	Number of 1 Kbyte blocks of I/O space occupied by this module (1 character)
21		20	Major functional revision level with leading blank (if single digit)
23	1	31	
25	1	30	Minor functional revision level with trailing blank (if single digit)
27		20	
29	Undefined	00	Manufacturer-dependent information, reserved for future use
2B	Undefined	00	
2D	Undefined	00	
2F	Undefined	00	
31	Undefined	00	
33	Undefined	00	
35	Undefined	00	
37	Undefined	00	
39	Undefined	00	
3B	Undefined	00	
3D	Undefined	00	
3F	Undefined	00	

The module has been designed so that it is only necessary to use odd backplane addresses to access the I.D. data. Thus, each of the 32 bytes of ASCII data have been assigned to the first 32 odd I/O Interface Block bytes (i.e., odd bytes 1 H-3FH).

I.D. information can be accessed simply by addressing the module base, offset by the specific address for the character(s) needed. For example, if the base address of the board is jumpered to 1000H,and if you wish to access the module model number (I/O interface block locations 11H, 13H, 15H, 17H, 19H, 1BH, and 1DH), you will individually add the offset addresses to the base addresses to read the hex-coded ASCII value at each location. In this example, the ASCII values which make up the module model number are found sequentially at locations 1011H, 1013H, 1015H, 1017H, 1019H, 101BH, and 101DH within the system's short I/O address space.

All XVME intelligent I/O modules are designed to perform diagnostic self-tests on power-up or reset. For non-intelligent modules, the user must provide the diagnostic program. The self-test provision allows the user to verify the operational status of a module by either visually inspecting the two LEDs (which are mounted on the front panel, as shown in Figure A-3), or by reading the module status byte (located at module base address + 81H).

Figure A-3 shows the location of the status LEDs on the module front panel . The two tables included with Figure A-3 define the visible LED states for the module test conditions on both the intelligent I/O modules and the non-intelligent I/O modules.



Status Bits 3	2	1	0	LEDs Green	Red	SYSFAIL*	Status
0	0	0	0	OFF	ON	ON	Module not yet tested
1	0	0	0	OFF	ON	ON	Module failed test
0	0	0	1	OFF	OFF	OFF	Inactive module
0	1	1	0	ON	ON	ON	Module undergoing test
1	1	1	1	ON	OFF	OFF	Module passed test
all others				X	X	X	Invalid and undefined

INTELLIGENT MODULE STATUS

Status Bits 1	0	LEDs Green	Red	SYSFAIL*	Status
0	0	OFF	ON	ON	Module failed, or not yet tested
0	1	OFF	OFF	OFF	Inactive module
1	0	ON	ON	ON	Module undergoing test
1	1	ON	OFF	OFF	Module passed test

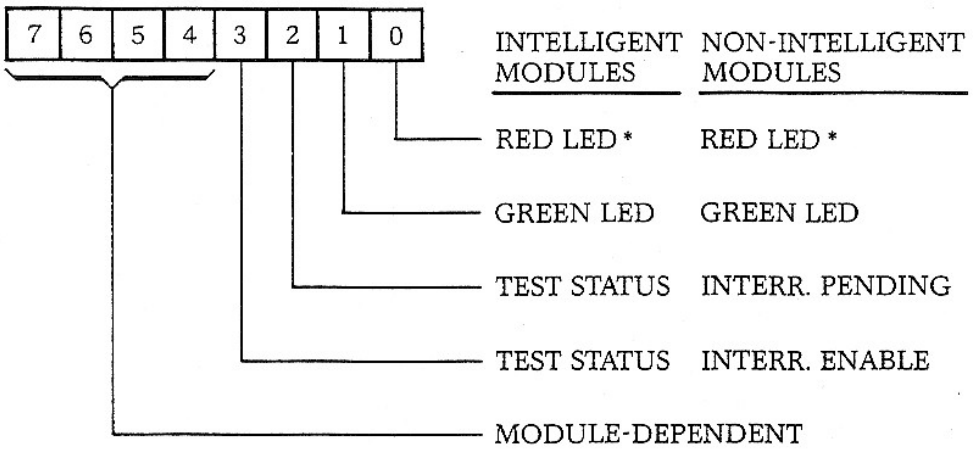
NON-INTELLIGENT MODULE STATUS

Figure A-3.
Module LED Status

The module status/control register (found at module base address + 81H) on intelligent XVME I/O modules provides the current status of the module self-test in conjunction with the current status of the front panel LEDs. The status register on intelligent modules is a read only register and it can be read by software to determine if the board is operating properly.

On non-intelligent XVME I/O modules, the status/control register is used to indicate the state of the front panel LEDs, and to set and verify module-generated interrupts. The LED status bits are read/write locations which provide the user with the indicators to accommodate diagnostic software. The Interrupt Enable bit is also a read/write location which must be written to in order to enable module-generated interrupts. The Interrupt Pending bit is a read only bit which indicates a module-generate depending interrupt.

Figure A-4 shows the status/control register bit definitions for both intelligent and non-intelligent XVME I/O modules.



Bit	Non-Intelligent Modules	Bit	Intelligent Modules
0	Read/Write - Red LED 0 = Red LED On 1 = Red LED Off	0	Read Only - Red LED 0 = Red LED On 1 = Red LED Off
1	Read/Write - Green LED 0 = Green LED Off 1 = Green LED On	1	Read Only - Green LED 0 = Green LED Off 1 = Green LED On
2	Read Only - Interrupt Pending 0 = No Interrupt 1 = Interrupt Pending	2 & 3	Read Only - Test Status Indicators Bit 3 Bit 2 0 0 = Self-test not started 0 1 = Self-test in progress 1 0 = Self-test failed 1 1 = Self-test passed
3	Read/Write - Interrupt Enable 0 = Interrupts Not Enabled 1 = Interrupts Enabled		
4	Module dependent	4	Module dependent
5	Module dependent	5	Module dependent
6	Module dependent	6	Module dependent
7	Module dependent	7	Module dependent

Figure A-4
Status Register Bit Definitions

INTERRUPT CONTROL

Interrupts for non-intelligent modules can be enabled or disabled by setting/clearing the Interrupt Enable bit in the module status register. The status pending on-board interrupts can also be read from this register. Interrupt control for intelligent modules is handled by the Interprocessor Communications Protocol (IPC).

Communications Between Processors

Communications between an intelligent "master" and an intelligent "slave" I/O module is governed by XYCOM's Interprocessor Communications Protocol. (IPC). This protocol involves the use of 20-byte Command Block data structures – located anywhere in the shared global RAM on an I/O module – to exchange commands and data between a host processor and an I/O module.

THE KERNEL

To standardize it XVME I/O Modules, XYCOM has designed them around "kernels" common from module to module. Each different module type consists of a standard kernel, combined with module-dependent application circuitry. Module standardization results in more efficient module design and allows the implementation of the Standard I/O Architecture. The biggest benefit of standardization for intelligent modules is that it allows the use of a common command language or protocol (Interprocessor Communications Protocol in this case).

The intelligent kernel is based around either a 68000 microprocessor or a 68B09 microprocessor (on the XVME-164 MBMM). This design provides the full complement of VMEbus Requester and Interpreter options for master/slave interfacing, as well as all of the advantages provided by the various facets of the XYCOM Standard I/O Architecture (as covered earlier in this Appendix).

The non-intelligent kernel provides the circuitry required to receive and generate all of the signals for a VMEbus defined 16-bit "slave" module. The non-intelligent kernel also employs the features of the XYCOM Standard I/O Architecture (as covered earlier in this Appendix).

Appendix B - **VMEbus CONNECTOR/PIN DESCRIPTIONS**

B.1 **INTRODUCTION**

The XVME-531 output module is a double-high (6U) VMEbus compatible module. On the rear edge of the board is a 96-pin bus connector labeled P1. The signals carried by connector P1 are the standard address, data, and control signals required for a P1 backplane interface, as defined by the VMEbus specification. Table B-1, on the following pages, identifies and defines the signals carried by the P1 connector. The second 96-pin bus connector, P2, is used for power and ground.

Table B-1. VMEbus Signal Identification

Signal Mnemonic	Connector and Pin Number	Signal Name and Description
ACFAIL*	1B:3	AC FAILURE: Open-collector driven signal which indicates that the AC input to the power supply is no longer being provided, or that the required input voltage levels are not being met.
IACKIN*	1A:21	INTERRUPT ACKNOWLEDGE IN: Totem-pole driven signal. IACKIN* and IACKOUT* signals form a daisy-chained acknowledge. The IACKIN* signal indicates to the VME board that an acknowledge cycle is in progress.
IACKOUT*	1A:22	INTERRUPT ACKNOWLEDGE OUT: Totem-pole driven signal. IACKIN* and IACKOUT* signals form a daisy-chained acknowledge. The IACKOUT* signal indicates to the next board that an acknowledge cycle is in progress.
AM0-AM5	1A:23 1B:16,17 18,19 1C:1	ADDRESS MODIFIER (bits 0-5): Three-state driven lines that provide additional information about the address bus, such as: size, cycle type, and/or DTB master identification.
AS*	1A:18	ADDRESS STROBE: Three-state driven signal that indicates a valid address is on the address bus.
A01-A23	1A:24-30 1C:15-30	ADDRESS BUS (bits 1-23): Three-state driven address lines that specify a memory address.
A24-A31	2B:4-11	ADDRESS BUS (bits 24-31): Three-state driven bus expansion address lines.
BBSY*	1B:1	BUS BUSY: Open-collector driven signal generated by the current DTB master to indicate that it is using the bus.
BCLR	1B:2	BUS CLEAR: Totem-pole driven signal generated by the bus arbitrator to request release by the DTB master if a higher level is requesting the bus.

Table B-1. VMEbus Signal Identification (Continued)

Signal Mnemonic	**Connector and Pin Number**	**Signal Name and Description**
BERR*	1C11:11	BUS ERROR: Open-collector driven signal generated by a slave. It indicates that an unrecoverable error has occurred and the bus cycle must be aborted.
BG0IN* BG3IN*	1B:4,6, 8,10	BUS GRANT (0-3) IN: Totem-pole driven dignals generated by the Arbiter or Requesters. Bus Grant In and Out signals form a daisy-chained bus grant. The Bus Grant In signal indicates to this board that it may become the next bus master.
BGOUT* - BG3OUT*	1B:5,7 9,11	BUS GRANT (0-3) OUT: Totem-pole driven signals generated by Requesters. These signals indicate that a DTB master in the dailsy-chain requires access to the bus.
BR0*-BR3*	1B:12-15	BUS REQUEST (0-3): Open-collector driven signals generated by Requesters. These signals indicate that DTB master in the daisy-chain requires access to the bus.
DS0*	1A:13	DATA STROBE 0: Three-state driven signal that indicates during byte and word transfers that a data transfer will occur ondata bus lines (D00-D07)
DS1*	1A:12	DATA STROBE 1: Three-state driven signal that indicates druing byte and word transfers that a data transfer will occur on data bus lines (D0-D15).
DTACK*	1A:16	DATA TRANSFER ACKNOWLEDGE: Open collector driven signal generated by a DTB slave. The falling edge of this signal indicates that valid data is available on the data bus during a read cycle, or that data has been accepted from the data bus during a write cycle.
D00-D15	1A:1-8 1C:1-8	DATA BUS (bits 0-15): Three-state driven, bi-directional data lines that provide a data path between the DTB master and slave.
GND	1A:9,11 15,17,19 1B:20,23, 1C:9 2B:2,12,22,31	GROUND

Table B-1. VMEbus Signal Identification (Continued)

Signal Mnemonic	Connector and Pin Number	Signal Name and Description
IACK*	1A:20	DATA TRANSFER ACKNOWLEDGE: Open-collector or three-state driven signal from any master processing an interrupt request. It is routed via the backplane to slot 1, where it is looped-back to become slot 1 IACKIN* in order to start the interrupt acknowledge daisy-chain.
IRQ1* IRQ7*	1B:24-30	INTERRUPT REQUEST (1-7): Open-collector driven signals, generated by an interrupter, which carry prioritized interrupt requests. Level seven in the highest priority.
LWORD*	1C:13	LONGWORD: Three-state driven signal indicates that the current transfer is a 32-bit transfer.
(RESERVED)	2B:3	RESERVED: Signal line reserved for future VMEbus enhancements. This line must not be used.
SERCLK	1B:21	A reserved signal which will be used as the clock for a serial communication bus protocol which is still being finalized.
SERDAT	1B:22	A reserved signal which will be used as the transmission line for serial communication bus messages.
SYSCLK	1A:10	SYSTEM CLOCK: A constant 16-MHz clock signal that is independent of processor speed or timing. It is used for general system timing use.
SYSFAIL*	1C:10	SYSTEM FAIL: Open-collector driven signal that indicates that a failure has occurred in the system. It may be generated by any module on the VMEbus.
SYSRESET*	1C:12	SYSTEM RESET: Open-collector driven signal which, when low, will cause the system to be reset.
WRITE*	1A:14	WRITE: Three-state driven signal that specifies the data transfer cycle in progress to be either read or written. A high level indicates a read operation, a low level indicates a write operation.

Table B-1. VMEbus Signal Identification (Continued)

Signal Mnemonic	**Connector and Pin Number**	**Signal Name and Description**
+5V STDBY	1B:31	+5 VDC STANDBY: This line supplies +5 VDC to devices requiring battery backup.
+5V	1A:32 1B:32 1C:32 2B:1,13,32	+5 VDC POWER: Used by system logic circuits.
+12V	1C:31	+12 VDC POWER: Used by system logic circuits.
-12V	1A:31	-12 VDC POWER: Used by system logic circuits

BACKPLANE CONNECTOR P1

The following table lists the P1 pin assignments by pin number order. (The connector consists of three rows of pins labeled A, B, and C.)

Table B-2. P1 Pin Assignments

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK(1)	A17
22	IACKOUT*	SERDAT(1)	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5v STDBY	+12V
32	+5V	+5v	+5v

Table B-3. Power and Ground Signals

Signal Mnemonic	**Connector and Pin Number**	**Signal Name and Description**
GND	2B:2 2B:12 2B:22 2B:31	Ground
+5V	2B:1 2B:13 2B:32	+5 VDC POWER: Used by system logic circuits.

Appendix C

QUICK REFERENCE GUIDE

This appendix contains the following contents for easy reference:

- XVME-566 I/O Interface Block (Memory map; Figure 4-1)
- Status/Control Register Bits (Figure 4-2)
- Sequence RAM Bits (Figure 4-5)
- Jumper Options (Table 2-1)
- Switch Options (Table 2-2)
- Module Addressing Options (Table 2-4)
- Input Connector JK1 (Table 2-16)
- Module base Address List (Table 2-5)

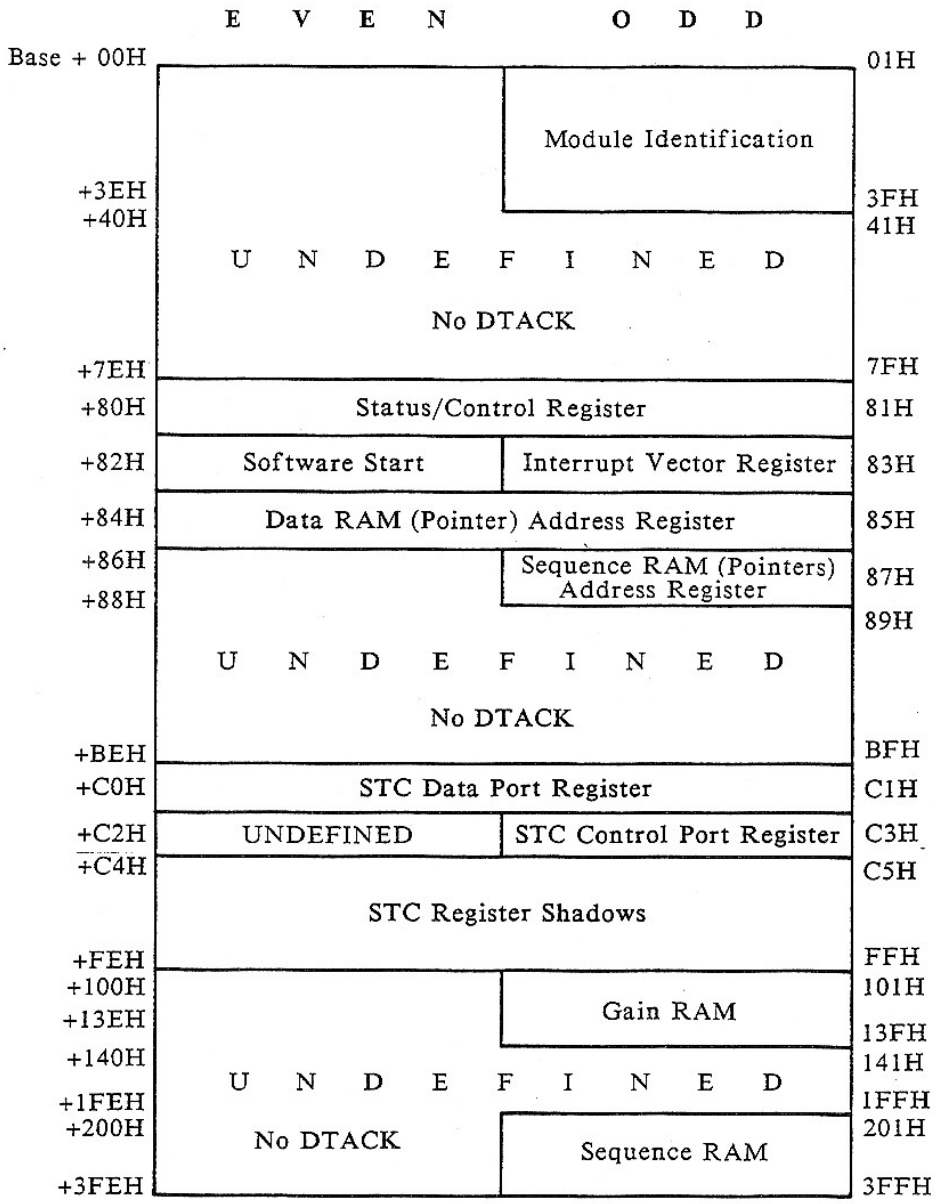


Figure 4-1
The XVME-566 Fast Analog Input Module I/O Interface Block

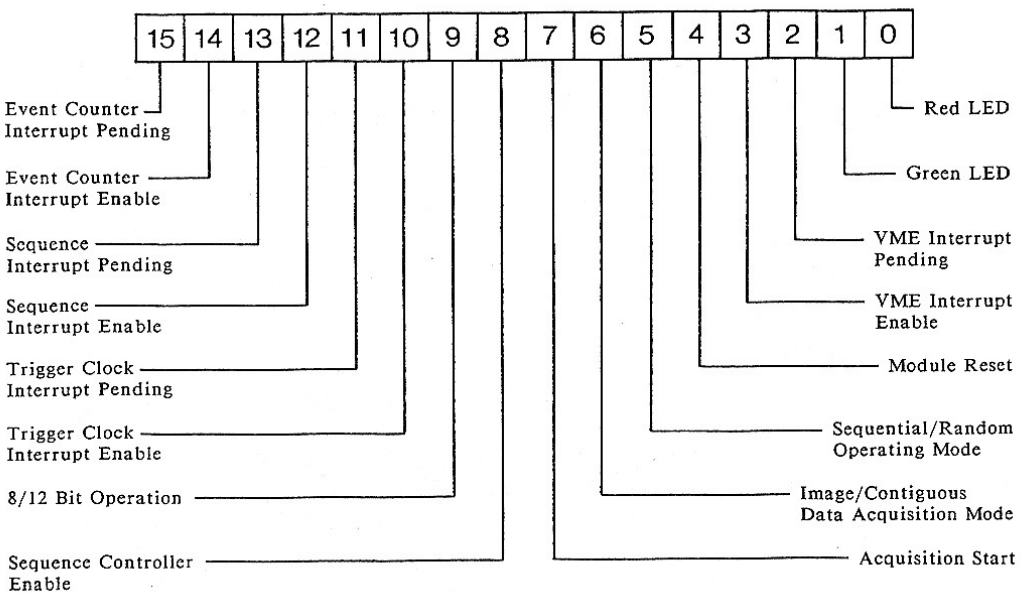


Figure 4-2

Status/Control Register

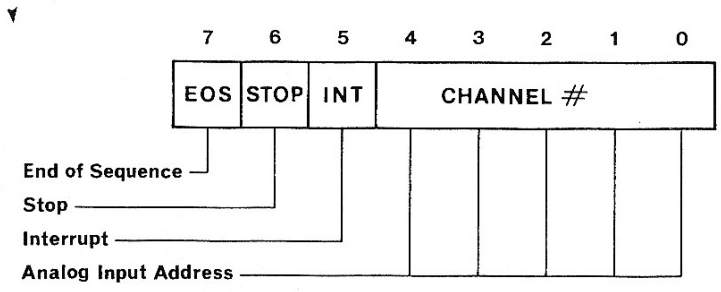


Figure 4-5

Sequence RAM

Table 2-1

XVME-566 Jumper Options

VMEbus OPTIONS	
Jumpers	**Use**
J1A,J1B,J2A,J2B	Enables, disenables IACKIN* to IACKOUT* daisy chain logic (See Section 2.6.4.2).
J3A,J3B	Standard or short I/O address select jumper (Section 2.6.2).
Analog-to-Digital Conversion OPTIONS	
Jumpers	**Use**
J4	Connects 8MHz clock to U54, Pin 1 for use with PAL device
J5,J16A,J16B	This group is used to select straight binary, offset binary and two's complement data format (Section 2.7.1).
J6	Configures data RAM size for 8K or 32K memory blocks (Section 2.6.1.2).
J7A,J7B,J7C,J7D,J7E,J7F	These jumpers are used to select one of three gain ranges for differential amplifier outputs (Section 2.7.4).

(continued on next page)

Table 2-1

XVME-566 Jumper Options (cont'd)

Analog-to-Digital Conversion Options (cont'd)	
Jumpers	**Uses**
J8A,J8B	Voltage range selector jumpers to analog-to-digital converter (Section 2.7.3).
J9A,J9B,J9C,J9D, J15	These jumpers provide the options of operating in single-ended, differential or pseudo-differential modes (Section 2.7.1.2).
J10A,J10B	These jumpers are used to configure inputs for either bipolar or unipolar input voltages and ranges (Section 2.7.3).
J11,J12,J14	These three jumpers are provided to allow for grounding of an input channel (in either the single[a]ended or differential mode) or for external ground reference for the external trigger (Sections 2.7.1.2, 2.7.2, 2.7.6).
J13A,J13B	Use of one of these jumpers will allow provisions for either on-board or off-board external triggers (Section 2.7.6).

Table 2-2
XVME-566 Switch Options

Switch Bank S1	
Switches	**Use**
Switches 1-3	Interrupt level select for any interrupts generated by the module (Section 2.6.4.1).
Switch Bank S2	
Switches	**Use**
Switches 1-8	Data RAM base address select in standard address space (Section 2.6.1.1).
Switch Bank S3	
Switches	**Use**
Switches 1-6	Module I/O base address select (Section 2.6.1)
Switch 7	This switch determines whether the module operates with address modifiers for the short I/O address space, or for those within the standard address space (Section 2.6.3).
Switch 8	This switch determines whether the module will respond to only supervisory accesses, or to both supervisory and non-privileged accesses (See Section 2.6.3).

Table 2-4
I/O Module Addressing Modifier Options

Jumper	**Switch 7 (S3)**	**Option Selected**
J3A	Open	Standard Data Access Operation
J3B	Closed	Short I/O Access Operation

Table 2-16 Input Connector JK1

Flat Cable Conductor	Single-Ended Configuration	Differential Configuration
1	CH. 0	CH. 0 LO
2	CH. 8	CH. 0 HI
3	ANALOG GND	ANALOG GND
4	CH. 9	CH. 1 HI
5	CH. 1	CH. 1 LO
6	ANALOG GND	ANALOG GND
7	CH. 2	CH. 2 LO
8	CH. 10	CH. 2 HI
9	ANALOG GND	ANALOG GND
10	CH. 11	CH. 3 HI
11	CH. 3	CH. 3 LO
12	ANALOG GND	ANALOG GND
13	CH. 4	CH. 4 LO
14	CH. 12	CH. 4 HI
15	ANALOG GND	ANALOG GND
16	CH. 5	CH. 5 HI
17	CH. 13	CH. 5 LO
18	ANALOG GND	ANALOG GND
19	CH. 6	CH. 6 LO
20	CH. 14	CH. 6 HI
21	ANALOG GND	ANALOG GND
22	CH. 15	CH. 7 HI
23	CH. 7	CH. 7 LO
24	ANALOG GND	ANALOG GND
25	CH. 16	CH. 8 LO
26	CH. 24	CH. 8 HI
27	ANALOG GND	ANALOG GND
28	CH. 25	CH. 9 HI
29	CH. 17	CH. 9 LO
30	ANALOG GND	ANALOG GND
31	CH. 18	CH. 10 LO
32	CH. 26	CH. 10 HI
33	ANALOG GND	ANALOG GND
34	CH. 27	CH. 11 HI
35	CH. 19	CH. 11 LO
36	ANALOG GND	ANALOG GND
37	CH. 20	CH. 12 LO
38	CH. 28	CH. 12 HI
39	ANALOG GND	ANALOG GND

(continued on next page)

Table 2-16

Input Connector JK1 (cont'd)

Flat Cable Conductor	**Single-Ended Configuration**	**Differential Configuration**
40	CH. 29	CH. 13 HI
41	CH. 21	CH. 13 LO
42	ANALOG GND	ANALOG GND
43	CH. 22	CH. 14 LO
44	CH. 30	CH. 14 HI
45	ANALOG GND	ANALOG GND
46	CH. 31	CH. 15 HI
47	CH. 23	CH. 15 LO
48	ANALOG GND	ANALOG GND
49	GND (EXT TRIG,PDI)	GND (EXT TRIG,PDI)
50	EXT TRIGGER	EXT TRIGGER

NOTE
When performing conversions in Sequence Mode, a software reset will lock the board. Before doing a software reset in this mode, bit 8 of the Status/Control register must be set to "0" (Sequence Controller Enable).

Table 2-3
Base Address Switch Options

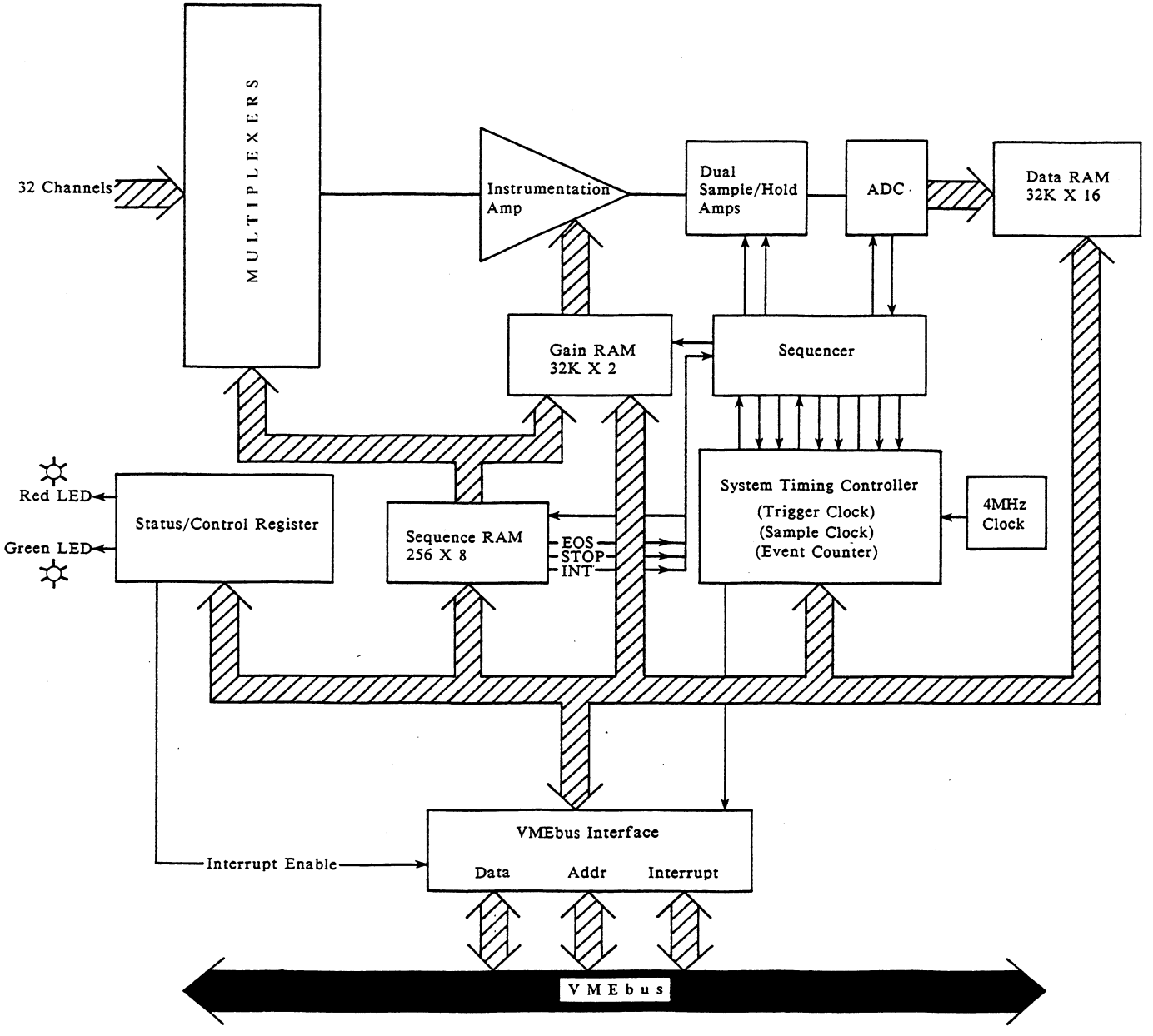
Switches						VME base address in VME Short I/O Address space
1(A15)	2(A14)	3(A13)	4(A12)	5(A11)	6(A10)	
0	0	0	0	0	0	0000H
0	0	0	0	0	1	0400H
0	0	0	0	1	0	0800H
0	0	0	0	1	1	0C00H
0	0	0	1	0	0	1000H
0	0	0	1	0	1	1400H
0	0	0	1	1	0	1800H
0	0	0	1	1	1	1C00H
0	0	1	0	0	0	2000H
0	0	1	0	0	1	2400H
0	0	1	0	1	0	2800H
0	0	1	0	1	1	2C00H
0	0	1	1	0	0	3000H
0	0	1	1	0	1	3400H
0	0	1	1	1	0	3800H
0	0	1	1	1	1	3C00H
0	1	0	0	0	0	4000H
0	1	0	0	0	1	4400H
0	1	0	0	1	0	4800H
0	1	0	0	1	1	4C00H
0	1	0	1	0	0	5000H
0	1	0	1	0	1	5400H
0	1	0	1	1	0	5800H
0	1	0	1	1	1	5C00H
0	1	1	0	0	0	6000H
0	1	1	0	0	1	6400H
0	1	1	0	1	0	6800H
0	1	1	0	1	1	6C00H
0	1	1	1	0	0	7000H
0	1	1	1	0	1	7400H
0	1	1	1	1	0	7800H
0	1	1	1	1	1	7C00H
1	0	0	0	0	0	8000H
1	0	0	0	0	1	8400H
1	0	0	0	1	0	8800H
1	0	0	0	1	1	8C00H
1	0	0	1	0	0	9000H
1	0	0	1	0	1	9400H
1	0	0	1	1	0	9800H
1	0	0	1	1	1	9C00H
1	0	1	0	0	0	A000H
1	0	1	0	0	1	A400H
1	0	1	0	1	0	A800H
1	0	1	0	1	1	AC00H
1	0	1	1	0	0	B000H
1	0	1	1	0	1	B400H
1	0	1	1	1	0	B800H
1	0	1	1	1	1	BC00H
1	1	0	0	0	0	C000H
1	1	0	0	0	1	C400H
1	1	0	0	1	0	C800H
1	1	0	0	1	1	CC00H
1	1	0	1	0	0	D000H
1	1	0	1	0	1	D400H
1	1	0	1	1	0	D800H
1	1	0	1	1	1	DC00H
1	1	1	0	0	0	E000H
1	1	1	0	0	1	E400H
1	1	1	0	1	0	E800H
1	1	1	0	1	1	EC00H
1	1	1	1	0	0	F000H
1	1	1	1	0	1	F400H
1	1	1	1	1	0	F800H
1	1	1	1	1	1	FC00H

NOTE

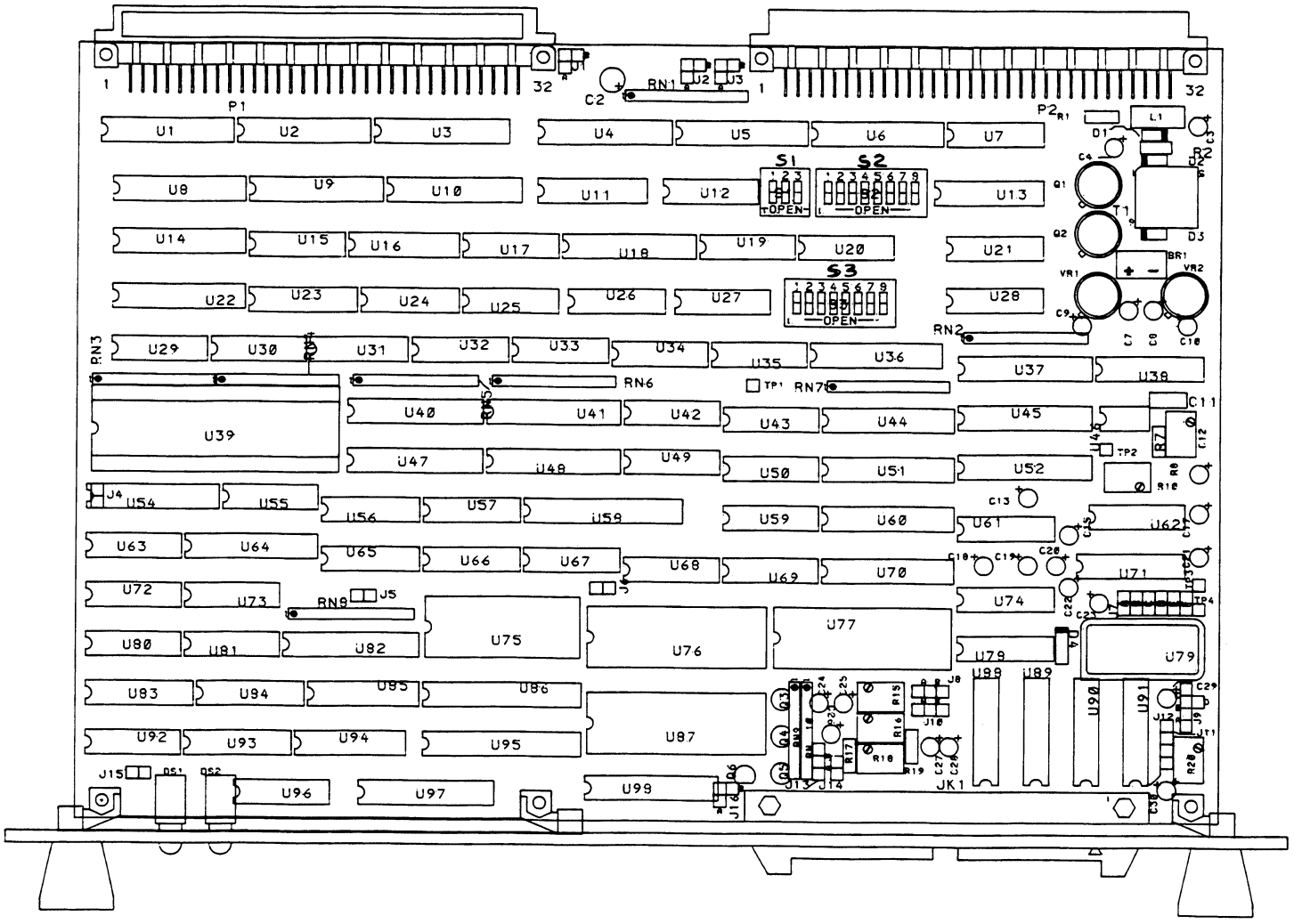
Open = Logic "1"
Closed = Logic "0"

Appendix D

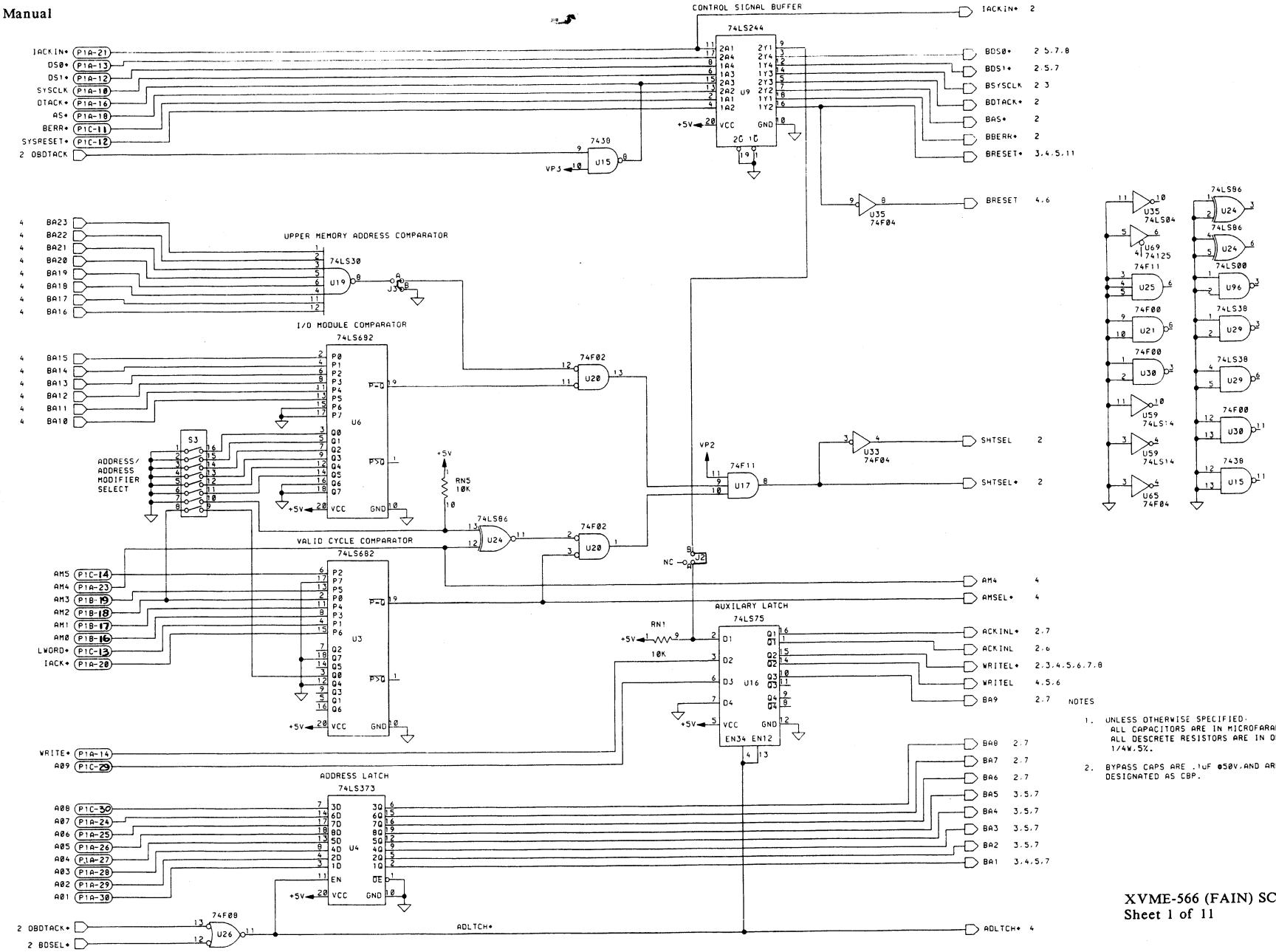
BLOCK DIAGRAM, ASSEMBLY DRAWING & SCHEMATICS



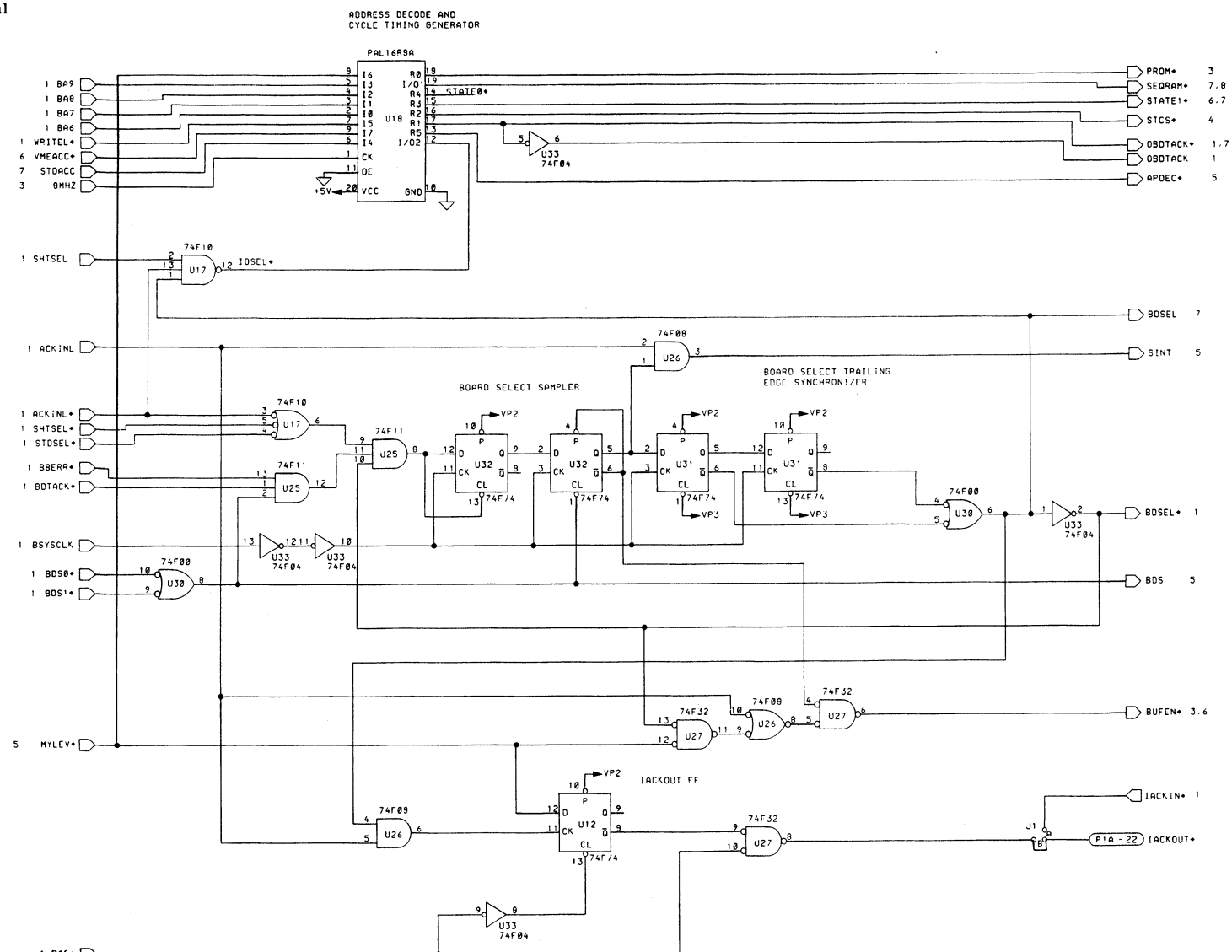
Block Diagram



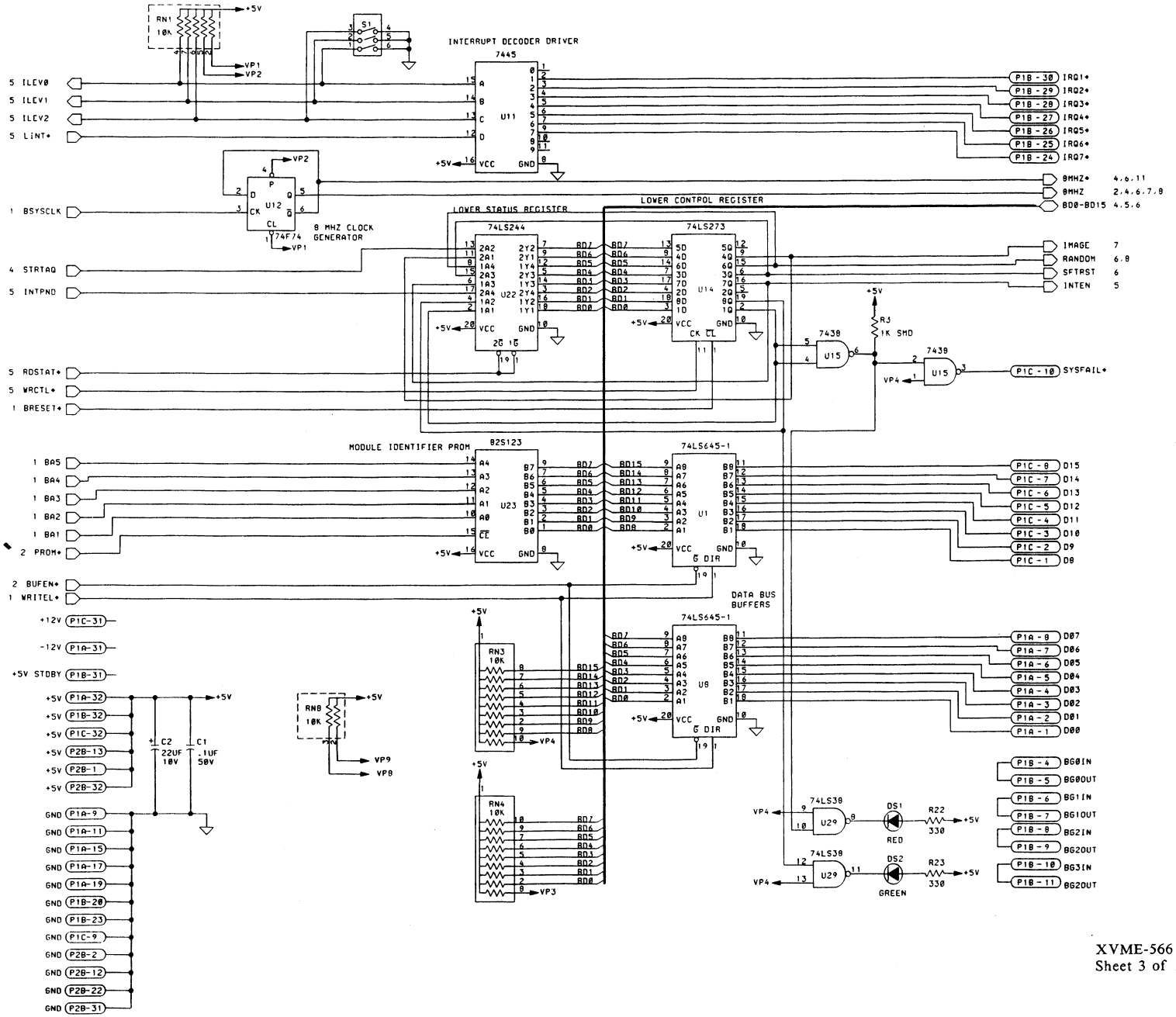
Assembly Drawing

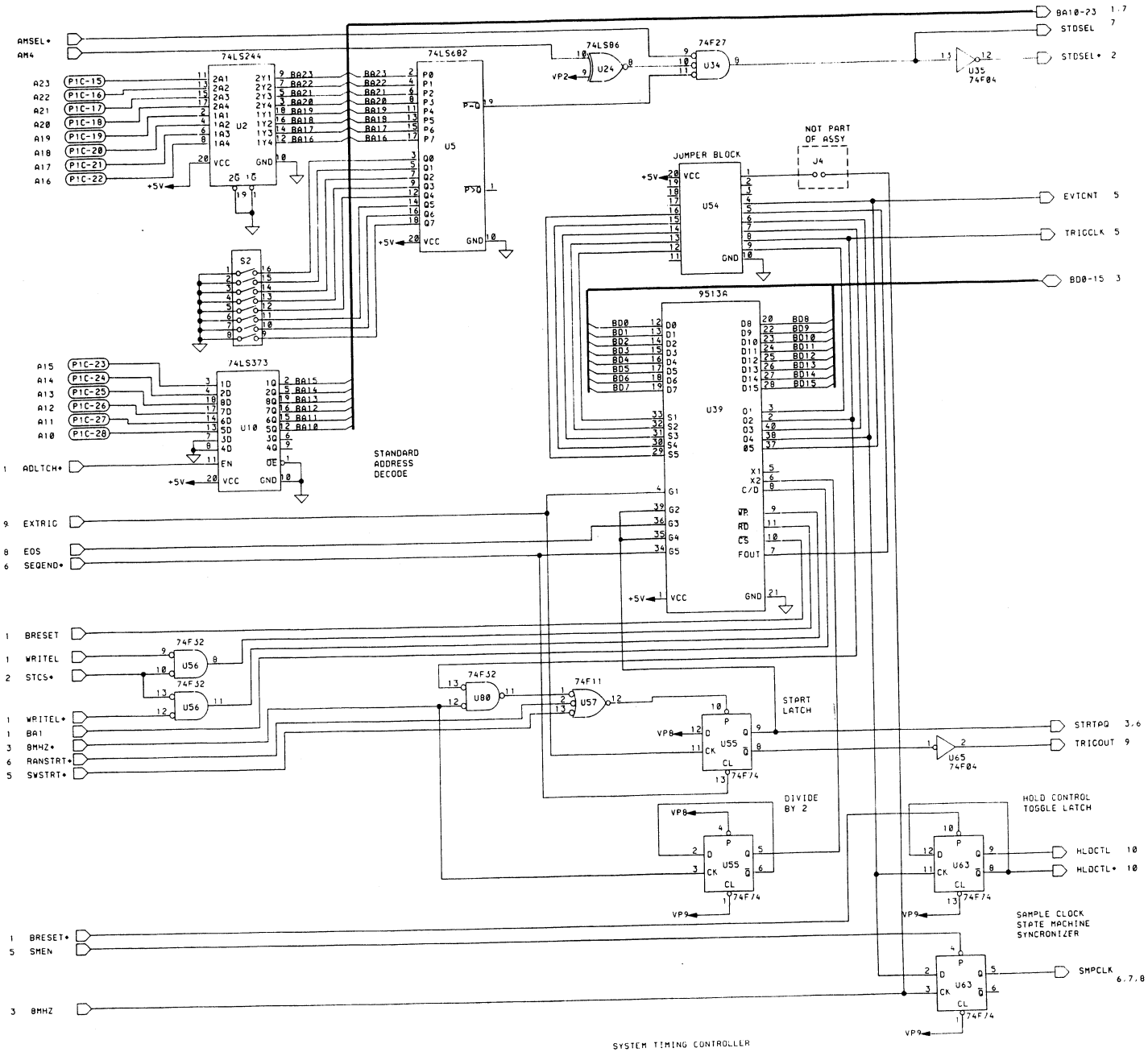


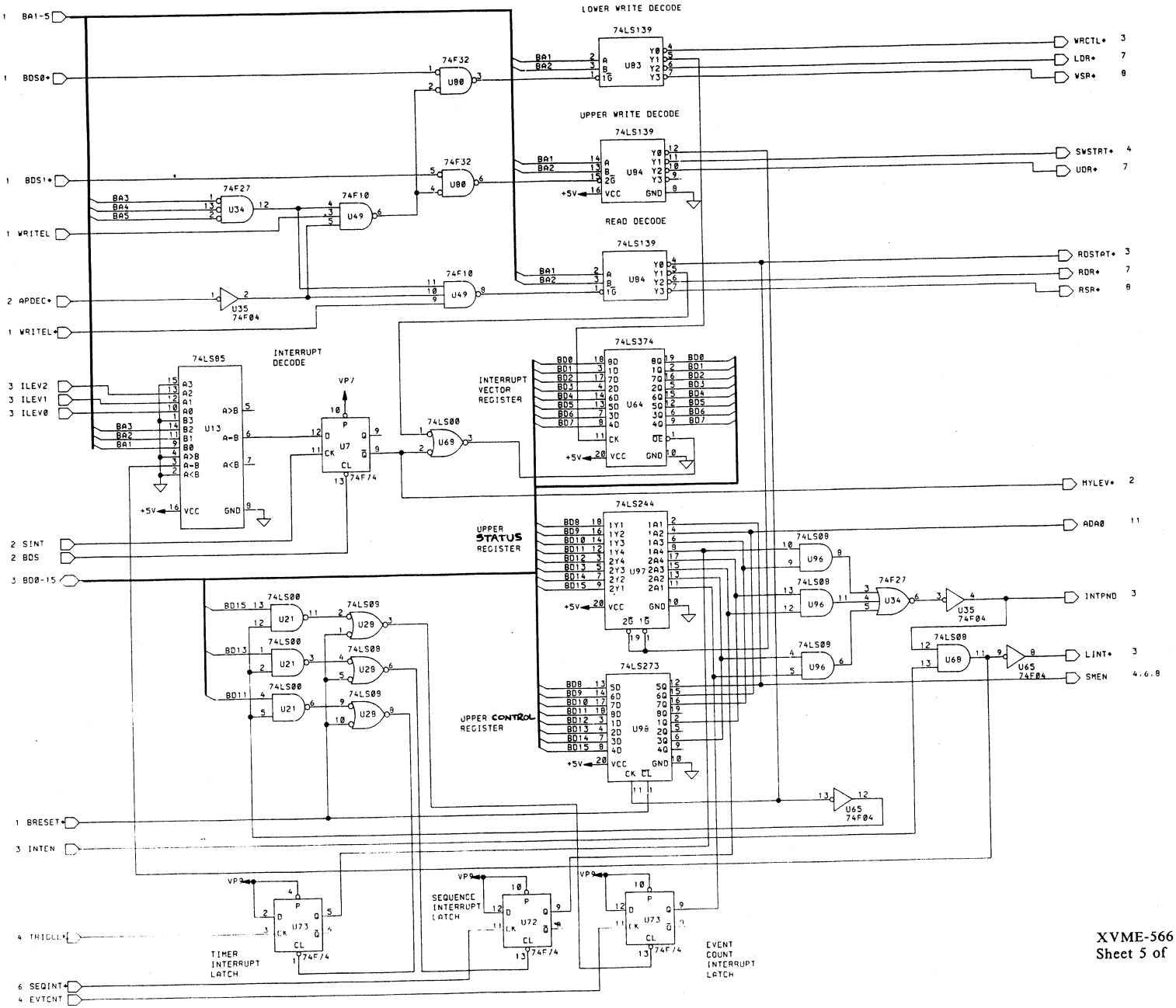
XVME-566 (FAIN) SCHEMATIC
Sheet 1 of 11



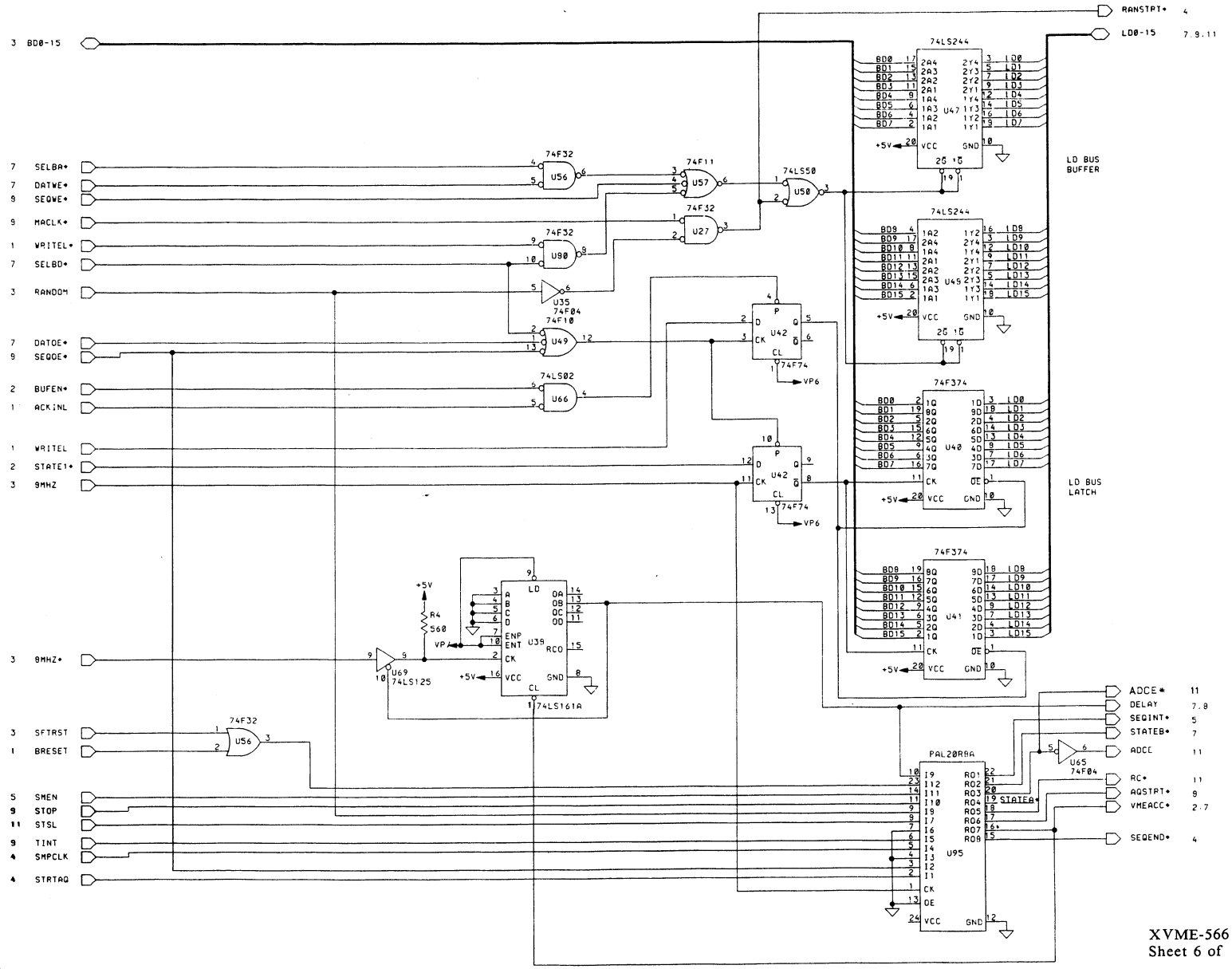
XVME-566 (FAIN) SCHEMATIC
Sheet 2 of 11


INTERRUPT DECODER DRIVER
7445
LOWER STATUS REGISTER
LOWER CONTROL REGISTER
74LS244
74LS273
8 MHZ CLOCK GENERATOR
MODULE IDENTIFIER PROM
82S123
74LS645-1
DATA BUS BUFFERS
74LS38
RED
GREEN
SYSFAIL*
IRQ1*
IRQ2*
IRQ3*
IRQ4*
IRQ5*
IRQ6*
IRQ7*
8MHZ*
8MHZ
BD0-BD15
IMAGE
RANDOM
SFTRST
INTEN
D15
D14
D13
D12
D11
D10
D9
D8
D07
D06
D05
D04
D03
D02
D01
D00
BG0IN
BG0OUT
BG1IN
BG1OUT
BG2IN
BG2OUT
BG3IN
BG3OUT

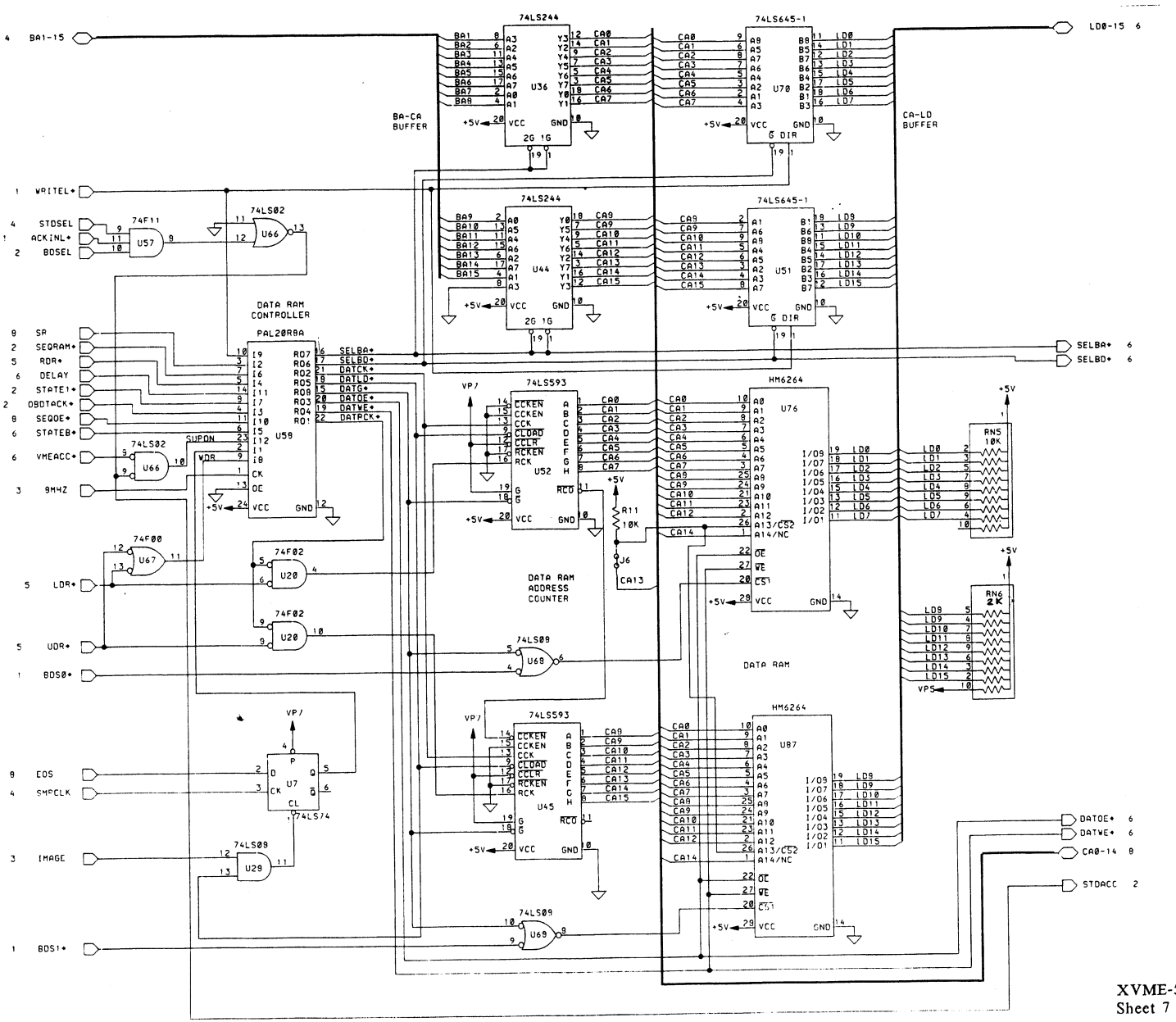

AMSEL*
AM4
74LS244
74LS682
74LS86
74F27
U24
U34
U35
74F04
BA10-23
STDSEL
STDSEL*
P1C-15
P1C-16
P1C-17
P1C-18
P1C-19
P1C-20
P1C-21
P1C-22
A23
A22
A21
A20
A19
A18
A17
A16
U2
U5
S2
JUMPER BLOCK
NOT PART OF ASSY
J4
U54
EVTCNT
TRIGCLK
BD0-15
9513A
U39
74LS373
U10
P1C-23
P1C-24
P1C-25
P1C-26
P1C-27
P1C-28
A15
A14
A13
A12
A11
A10
ADLTCH*
STANDARD ADDRESS DECODE
EXTRIG
EOS
SEQEND*
BRESET
WRITEL
STCS*
WRITEL*
BA1
8MHZ*
RANSTRT*
SWSTRT*
74F32
U56
U80
74F11
U57
START LATCH
U55
74F74
STRTPQ
TRIGOUT
U65
DIVIDE BY 2
HOLD CONTROL TOGGLE LATCH
U63
HLDCTL
HLDCTL*
SAMPLE CLOCK STATE MACHINE SYNCRONIZER
SMPCLK
BRESET*
SMEN
8MHZ
VP2
VP8
VP9
+5V
GND
SYSTEM TIMING CONTROLLER

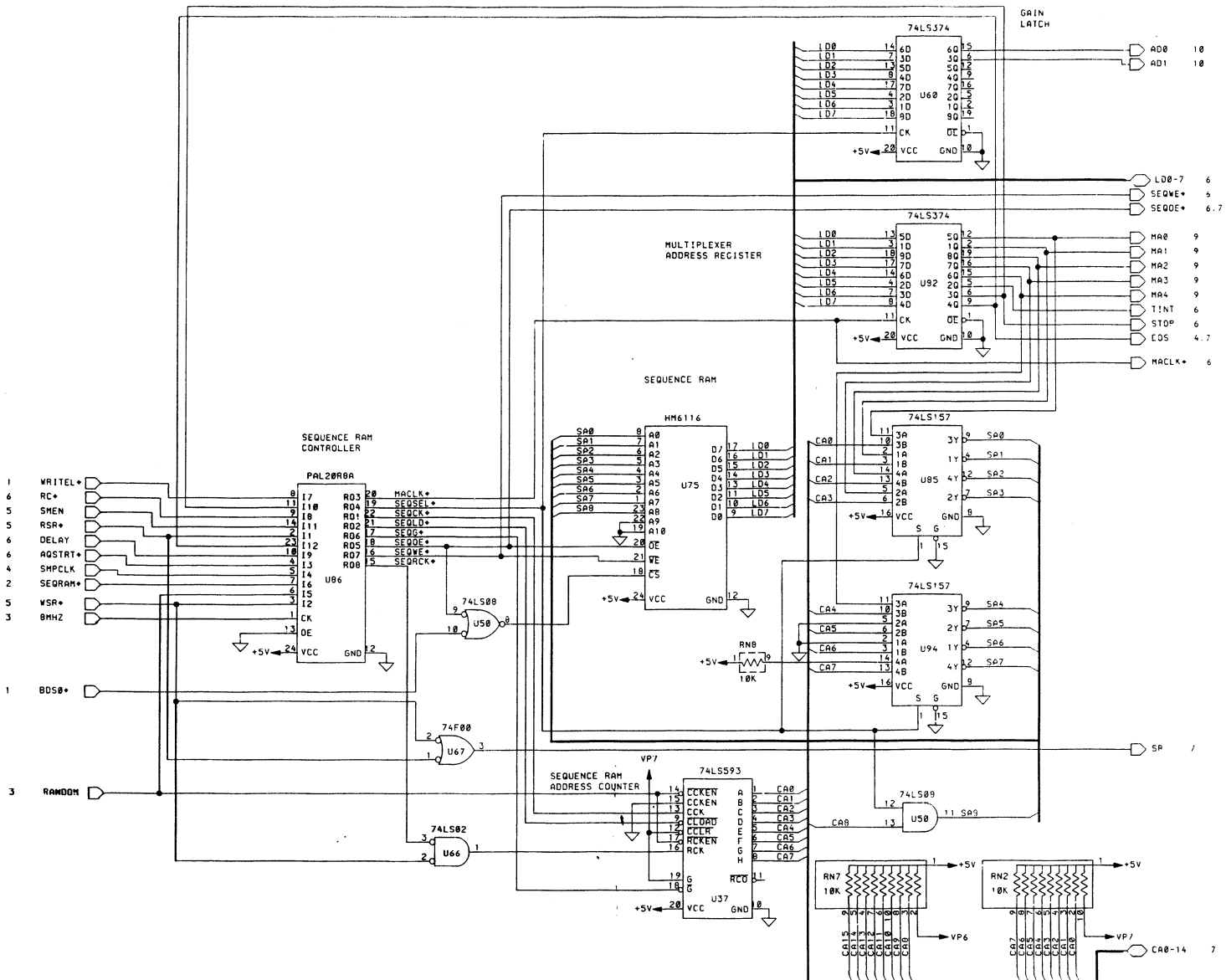


XVME-566 (FAIN) SCHEMATIC
Sheet 5 of 11

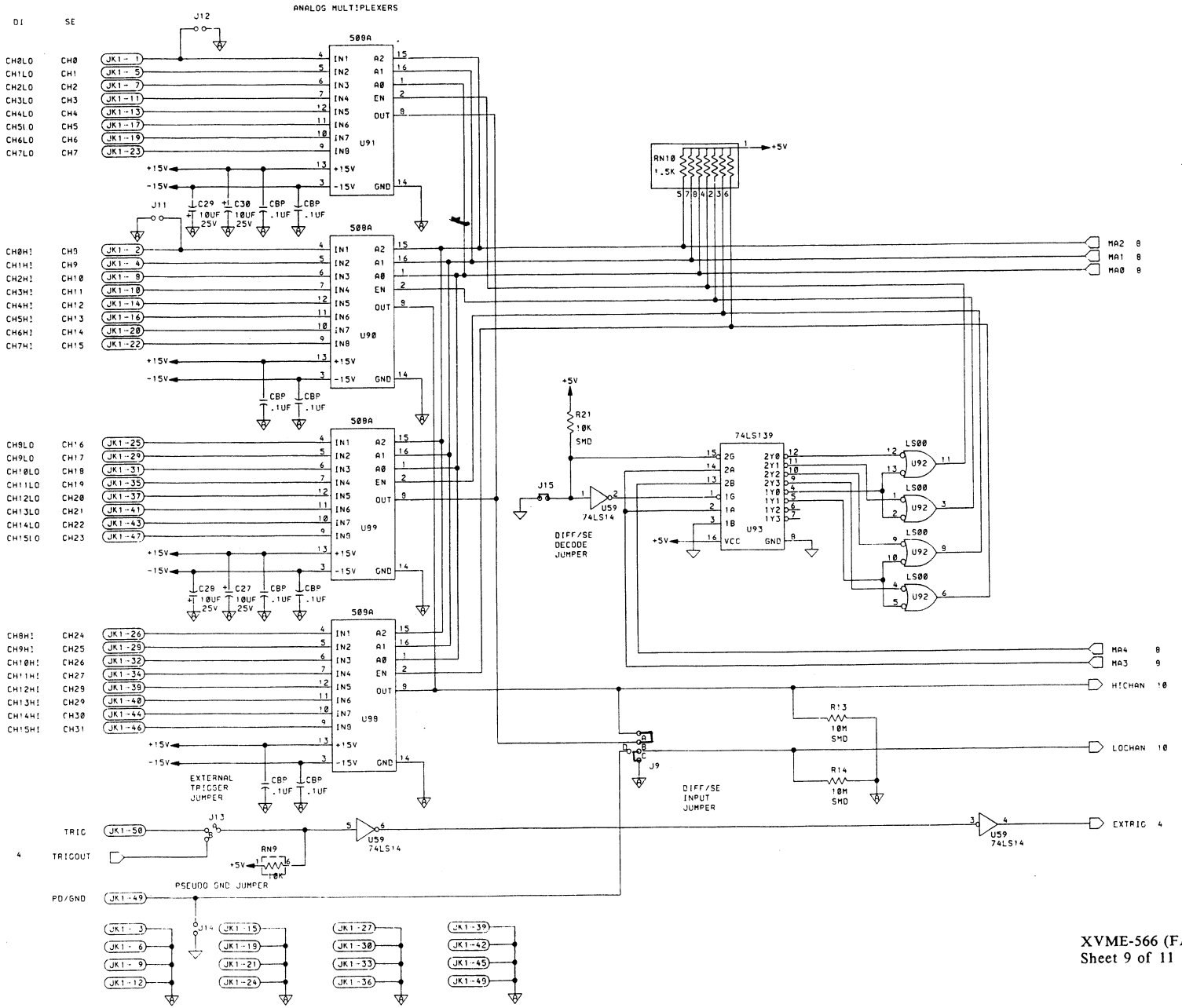


XVME-566 (FAIN) SCHEMATIC
Sheet 6 of 11

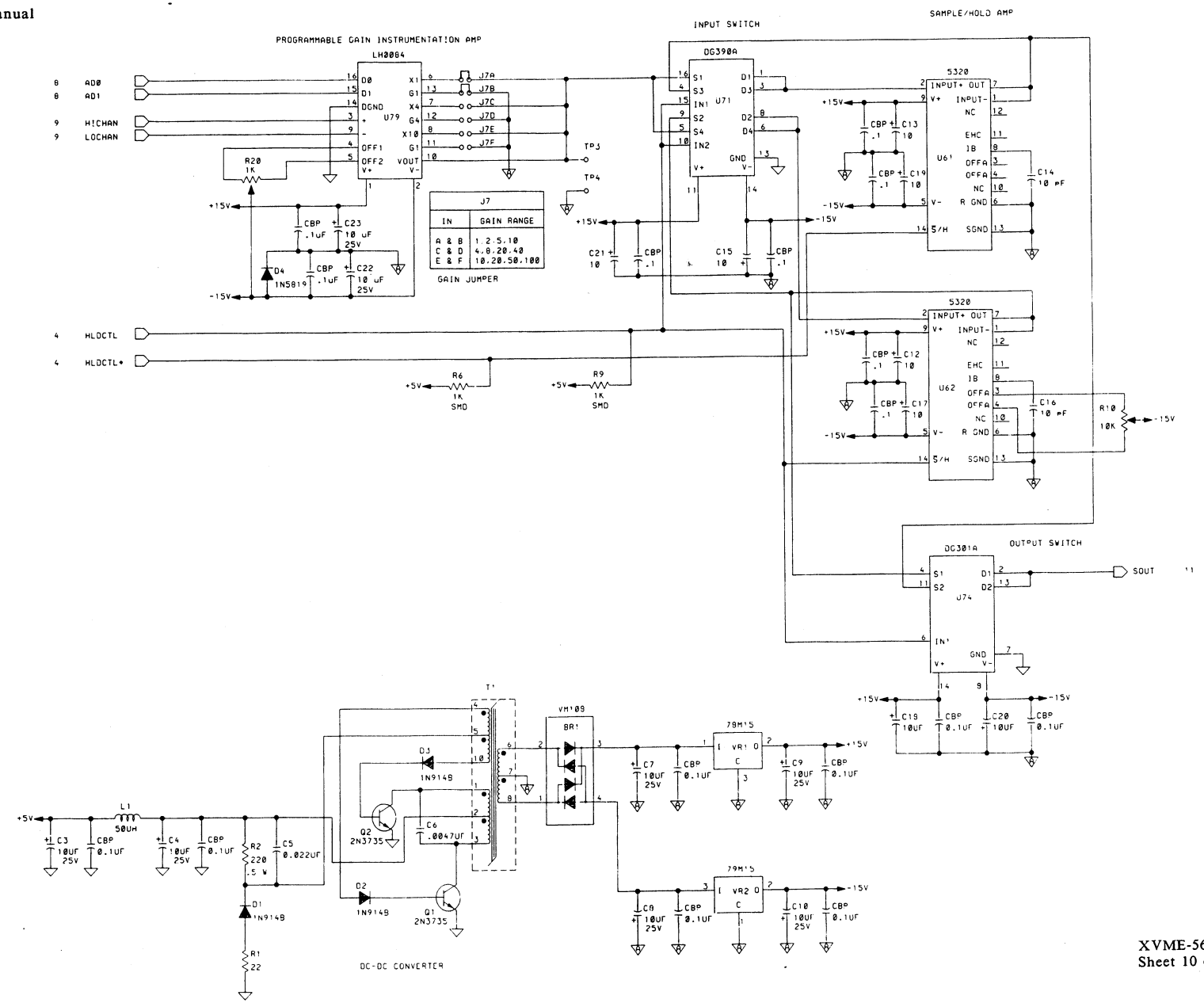

BA1-15
WRITEL*
STDSEL
ACKINL*
BDSEL
74F11
U57
74LS02
U66
74LS244
U36
BA-CA BUFFER
74LS645-1
U70
CA-LD BUFFER
LD0-15
74LS244
U44
74LS645-1
U51
DATA RAM CONTROLLER
PAL20R8A
U59
SR
SEQRAM*
RDR*
DELAY
STATE1*
OBDTACK*
SEQOE*
STATEB*
VMEACC*
9MHZ
SUPDN
WDR
SELBA*
SELBD*
74LS593
U52
HM6264
U76
RN5 10K
RN6 2K
R11 10K
CA13
DATA RAM ADDRESS COUNTER
DATA RAM
74F00
U67
74F02
U20
74LS08
U68
LDR*
UDR*
BDS0*
VP7
74F74
U7
EOS
SMPCLK
IMAGE
U29
74LS593
U45
HM6264
U87
74LS09
U69
BDS1*
DATOE*
DATWE*
CA0-14
STDACC

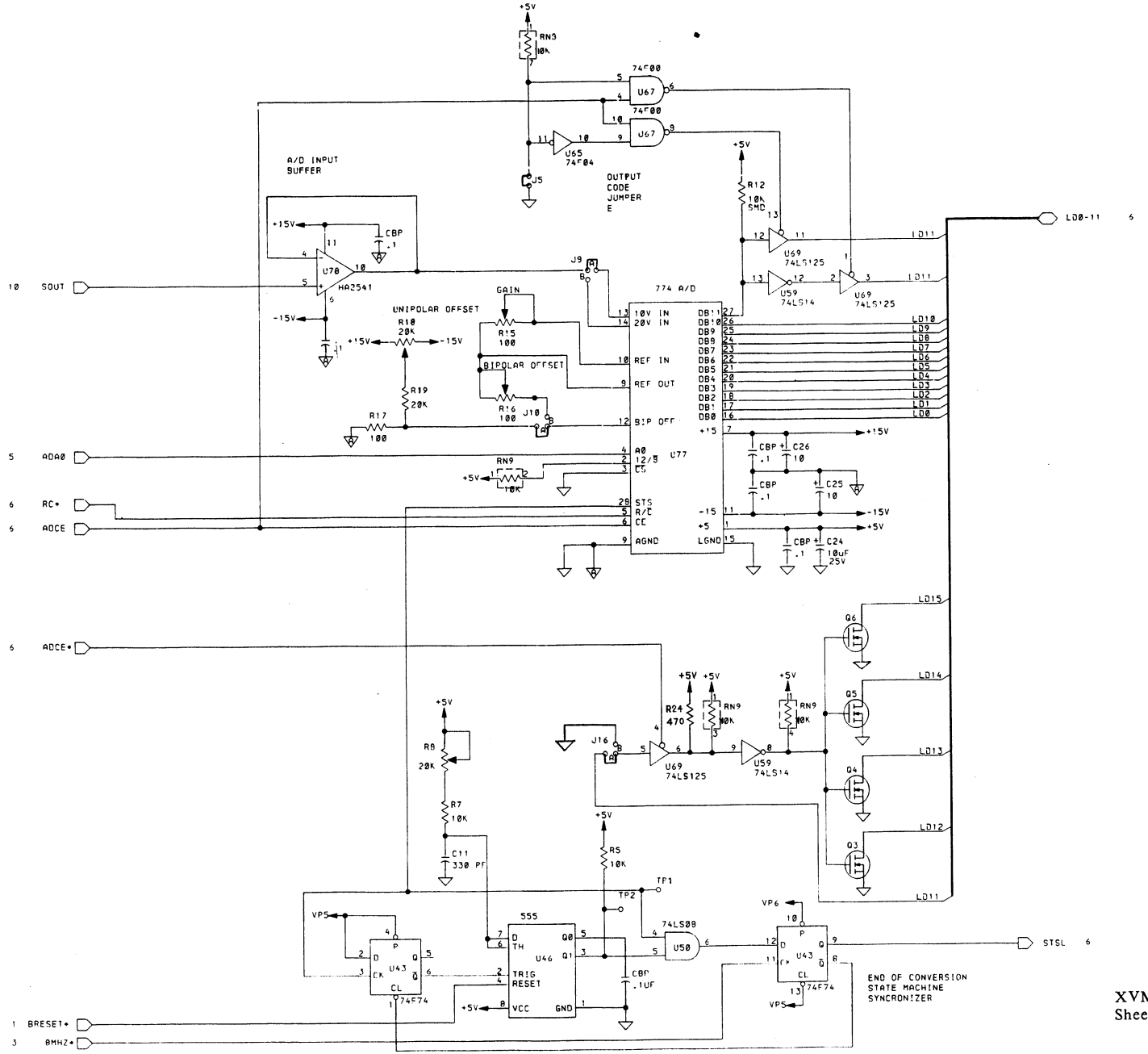


XVME-566 (FAIN) SCHEMATIC
Sheet 8 of 11



XVME-566 (FAIN) SCHEMATIC
Sheet 9 of 11

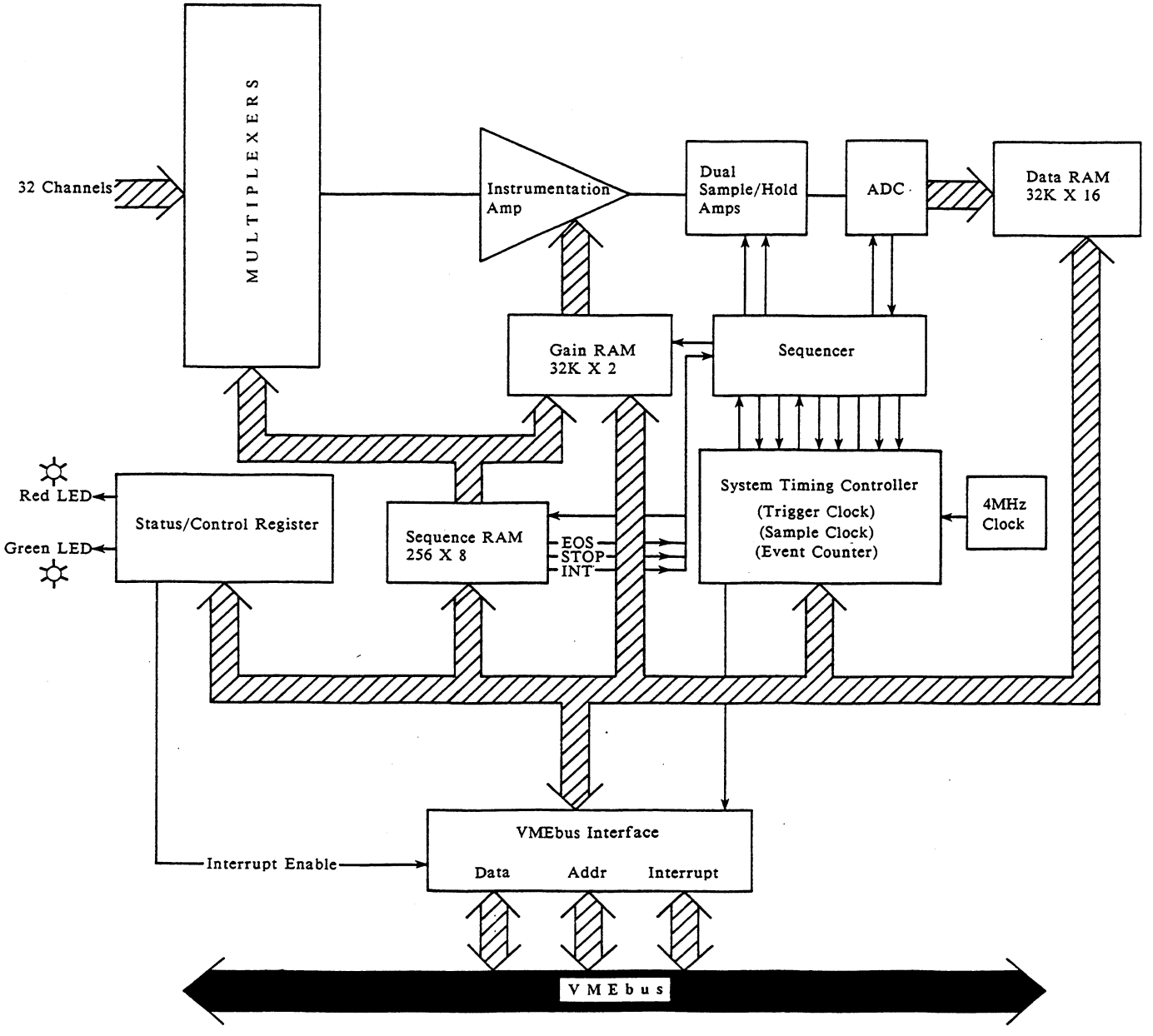

PROGRAMMABLE GAIN INSTRUMENTATION AMP
LH0084
U79
INPUT SWITCH
DG390A
U71
SAMPLE/HOLD AMP
5320
U61
U62
OUTPUT SWITCH
DG301A
U74
J7
IN
GAIN RANGE
A & B
C & D
E & F
1.2.5.10
4.8.20.40
10.20.50.100
GAIN JUMPER
AD0
AD1
HICHAN
LOCHAN
HLDCTL
HLDCTL*
SOUT
DC-DC CONVERTER
VM109
79M15
1N5819
1N914B
2N3735



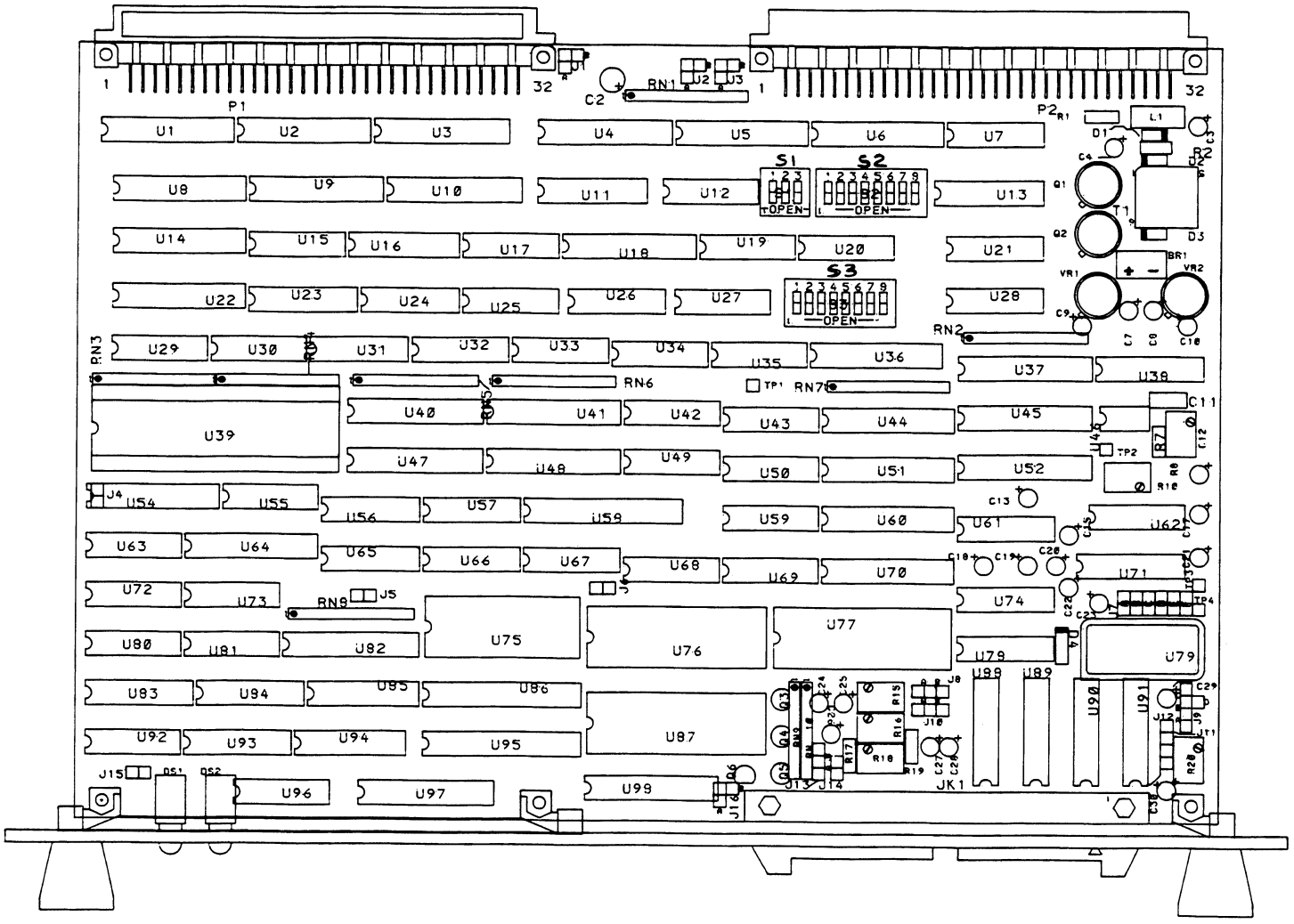
XVME-566 (FAIN) SCHEMATIC
Sheet 11 of 11

Appendix D

BLOCK DIAGRAM, ASSEMBLY DRAWING & SCHEMATICS



Block Diagram



Assembly Drawing