GE Fanuc
Intelligent Platforms

Hardware Reference

VME-4145

4-Channel 16-bit Arbitrary Waveform Generator Board with Autocalibration

First Edition

Part No: 500-024145-000 REV. A

Document History

Hardware Reference Manual Document Number: 500-024145-000 REV. A June 2, 2008

Table of Contents

List of Figures

List of Tables

• Overview

GE Fanuc Intelligent Platforms' VME-4145 is an Analog Output board that provides four high-quality analog output channels with 16-bit resolution. Each output has a dedicated Digital-to-Analog Converter (DAC), and can source or sink 10 mA ±10 V. Each channel has a dedicated 64 Kword waveform buffer. The buffer is scanned at a user-programmable rate. The buffer data is fed to its dedicated DAC. Loading appropriate data will cause the DAC's output to generate an arbitrary waveform. Each buffer may be segmented to provide independent subwaveforms. This segmentation allows switching between multiple output waveforms at high speeds. The analog outputs can be independently disconnected from the field wiring for offline testing.

Autocalibration is initiated by a VME software command. An onboard Digital Signal Processor (DSP) and a 20-bit Analog-to-Digital Converter (ADC) are used to calibrate the offsets and gains of the DACs. During calibration, a table of offset and gain coefficients are compiled and stored in RAM. There are entries for offset and gain corresponding to each of the four channels configured in each of the five output voltage ranges.

The unit is designed to download an arbitrary waveform and then replay it. Any arbitrary waveform may be programmed. The standard square, sine, triangle, and ramp are but a small sampling of the types of waveforms that may be generated. It is the user's responsibility to generate a table of hexadecimal data corresponding to the voltages required for their waveform. The appendix has several examples of C code programs which illustrate programming of the board. The VME-4145 contains a built-in square-wave generator command. This automatic waveform is useful to quickly determine proper operation of the unit. The user's software must download this waveform table from the VME host into the board's waveform buffer RAM memory. There are five separate 64 K x 16 RAM memories. The first is a dual-port RAM and is accessible by the VME host processor and onboard DSP processor. The user downloads an arbitrary waveform table into this common memory. During this download period, all four channels may be "playing-out" the information previously stored in their four dedicated RAM memories. Once the user completes downloading from the VME into the common RAM, the user instructs the DSP to move the common buffer data to one of the four output channels. At this point, the DSP must take the desired channel offline to update its RAM memory with the new waveform stored in the common memory. This block movement takes a minimum of 9.2 µs per sample; the larger the waveform table size, the longer it will take. During this block movement, the DSP will apply gain and offset corrections to the user's data if calibration is required.

To enable the VME-4145 to change from one waveform to another waveform in real time, the unit supports segmentation. The user may segment the 64 K waveform space into multiple smaller waveforms. A maximum of 64 unique waveforms, each up to 1,024 words in length, may be programmed. In real time, the user may switch between any one of these 64 waveforms or program an automatic seamless sequencing of any number of these. Each waveform is generated start to finish and then the next is seamlessly started. A link list software approach is used to program the sequencing of segments in real time by the host processor. All of the waveforms must be preloaded before waveform generation begins.

The VME-4145 supports an external trigger input. This is used to control the VME-4145 externally. A trigger event will cause the VME-4145 to start playback of a waveform. Before the waveform has completed one cycle through the table, the user may retrigger the unit. This will restart the waveform output from the beginning. This feature allows the unit to be synchronized to some external condition.

The VME-4145 supports sweeping of the output frequency. The board's internal sample rate clock (one of four) may be changed in real time while a waveform is being generated. This feature requires real-time VME host control interaction. The host may instruct the DSP to change the internal sample rate clock at any time. The board also supports external input of the sample clock. The external source may also sweep its frequency. Finally, the output clock's static logic (one or zero) state may be programmed by the host for extremely long output update rates. Channel zero's sample clock may also be used as a common clock to synchronize any or all of the remaining channels to the same rate. The fastest rate from one sample to the next is 400 ns, equivalent to 2.5 million samples per second.

Features

The VME-4145 features the following:

- Four-channel analog waveform generator
- Autocalibration of all channels to a single reference
- Continuous burst, single burst, or burst/idle/burst modes
- One 16-bit DAC per output
- Programmable sample clock for each channel, internal or external
- Up to 2.5 million samples/second
- External clock and trigger input per channel
- Variable length (up to 65,536 word) waveform buffer per channel
- Buffer segmentation for multiple waveforms in memory
- Unipolar 5 and 10 V ranges, Bipolar 2.5, 5, and 10 V ranges
- 10 mA output drive current per channel
- 1.5 Ω output impedance per channel
- Deglitched DAC outputs with optional 4-pole low pass filtering
- External SYNC output (per channel)
- Onboard DSP and 20-bit Analog-to-Digital Converter for automatic calibration and diagnostic self-test
- Field wiring disconnect for offline self-test diagnostics
- Front panel reference voltage(s) access
- Programmable SYNC pulse polarity
- Trigger input: Level HI/LO or rising/falling edge per channel.

Organization

This manual is composed of the following chapters and appendix:

Overview provides a general description of the VME-4145 and General Safety terms and symbols.

Chapter 1 Configuration and Installation describes unpacking, inspection, hardware jumper settings, connector definitions, installation, and system setup of the VME-4145.

Chapter 2 Theroy of Operation describes functionality, hardware settings, connector definitions, and operation of the VME-4145.

Chapter 3 Programming describes the sequence of initialization, defining waveforms and commands for wave generation, along with the coefficient registers for channels.

Maintenance provides GE Fanuc Intelligent Platforms' contact information relative to the care and maintenance of the unit.

Appendix A Sample Codes illustrates and defines the code samples.

References

For a detailed description and specification of the VMEbus, please refer to:

VMEbus Specification Rev. C. and the VMEbus Handbook

VMEbus International Trade Assoc. (VITA)
7825 East Gelding Dr.
Suite 104
Scottsdale, AZ 85260
(602) 951-8866
(602) 951-0720 (FAX)
www.vita.com

Safety Summary

The following general safety precautions must be observed during all phases of the operation, service and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture and intended use of this product.

GE Fanuc Intelligent Platforms assumes no liability for the customer's failure to comply with these requirements.

Ground the System

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

Do Not Operate in an Explosive Atmosphere

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

Keep Away from Live Circuits

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

Do Not Service or Adjust Alone

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

Do Not Substitute Parts or Modify System

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

Dangerous Procedure Warnings

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

WARNING

Dangerous voltages, capable of causing death, are present in this system.
Use extreme caution when handling, testing and adjusting.

Warnings, Cautions and Notes

WARNING

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

CAUTION

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

NOTE

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

TIP

Tip denotes a bit of expert information.

LINK

This is link text.

1 • Configuration and Installation

This chapter describes the hardware jumper settings, connector descriptions, installation, system setup and configuration of the VME-4145.

1.1 Unpacking Procedures

Any precautions found in the shipping container should be observed. All items should be carefully unpacked and thoroughly inspected for damage that might have occurred during shipment. The board(s) should be checked for broken components, damaged printed circuit board(s), heat damage and other visible contamination. All claims arising from shipping damage should be filed with the carrier and a complete report sent to GE Fanuc Intelligent Platforms Customer Care.

CAUTION

Some of the components assembled may be sensitive to electrostatic discharge and damage may occur on boards that are subjected to a high energy electrostatic field. When the board is placed on a bench for configuring, etc., it is suggested that conductive material be inserted under the board to provide a conductive shunt. Unused boards should be stored in the same protective boxes in which they were shipped.

1.2 Handling Precaution

Electronic assemblies use devices that are sensitive to static discharge. Observe anti-static procedures when handling these boards. All products should be in an anti-static plastic bag or conductive foam for storage or shipment. Work at an approved anti-static workstation when unpacking boards.

1.3 Hardware Setup

The VME-4145 is factory populated with user-specified options as part of the VME-4145 ordering information. Contact customer care to receive a Return Material Authorization (RMA). RMAs are available at rma@gefanuc.com

GE Fanuc Intelligent Platforms Customer Care is available at: 1-800-GEFANUC

(1-800-240-7782), 1-780-401-7700 or
E-mail us at support.embeddedsystems@gefanuc.com

The VME-4145 is tested for system operation and shipped with factory-installed header jumpers. The physical location of the jumpers and connectors for the VME-4145 are illustrated in Figure 1-1.

CAUTION

All jumpers marked *User Configurable* in the following tables may be changed or modified by the user. All jumpers marked *Factory Configured* should not be modified by the user.

Care must be taken when making jumper modifications to ensure against improper settings or connections. Improper settings may result in damage to the unit.

CAUTION

Modifying any jumper not marked *User Configurable* will void the Warranty and may damage the unit. The default jumper condition of the VME-4145 is expressed in Table 1-1.

1.4 Installation

CAUTION

Do not install or remove the boards while power is applied.

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

1.4.1 Before Applying Power: Checklist

Before installing the board in a VME system, check the following items to ensure that the board is ready for the intended application.

1. Verify that the sections pertaining to programming and configuration, Chapter 1 Configuration and Installation and Chapter 3 Programming, have been reviewed and applied to system requirements._____
2. Review Table 1-1 on page 19 and Table 1-2 on page 21 to verify that all jumpers are configured correctly for the application. _____
3. Verify that the I/O cables are properly terminated for the input/output connectors. Refer to Figure 1-3 for connector descriptions. _____
4. Ensure that all system cable connections are correct. _____
5. Calibration has been performed at the factory. If recalibration should be required, refer to “Calibration” on page 27. _____

After the checklist above has been completed, the board can be installed in a VME system. This board can be installed in any slot position, except slot one which is reserved for the system controller.

1.5 Operational Configuration

Control of the VME-4145 board address and I/O access mode are determined by field configurable, onboard jumpers. This section describes the use of those jumpers, and their effects on board performance. The locations and functions of all VME-4145 jumpers are shown in Figure 1-1 and Figure 1-2. For jumper settings refer to the Table 1-1 on page 19 and Table 1-2 on page 21.

Figure 1-1 VME-4145 Jumper and Test Point Locations

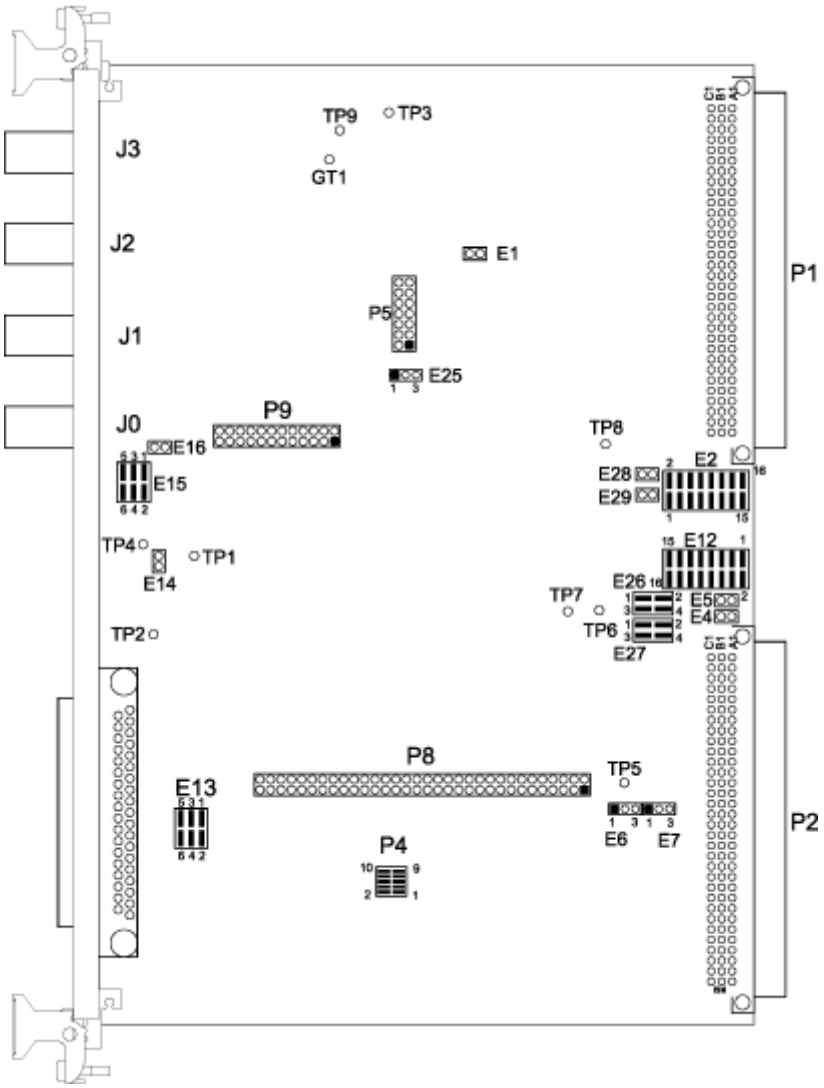


Table 1-1 Main Board Jumper Descriptions

Jumper	Pins	Description	Factory Default
E1	UIOC Control		
	1-2	Jumper Off - Logic 1 - No UIOC Control Jumper On - Logic 0 - UIOC Control	Omitted
E4	1-2	Install for External -15 VDC Power Supply	Omitted
E5	1-2	Install for External +15 VDC Power Supply	Omitted
E6	1-2	Local -15 V Power	Installed
	2-3	Remote -15 V Power	Omitted
E7	1-2	Local +15 V Power	Installed
	2-3	Remote +15 V Power	Omitted

Table 1-1 Main Board Jumper Descriptions (Continued)

Jumper	Pins	Description	Factory Default
E13 See Note 1 See Note 2	Channel 0 Output Offset Select		
	1-2	Install to Use Channel 1 DAC to Offset DC Zero Point of Channel 0 Output **DO NOT INSTALL 3-4 OR 5-6 WITH THIS OPTION, IT WILL SHORT OUT THE CHANNEL 1 DAC OUTPUT.**	Omitted
	3-4	Install to Use J0 BNC's Ground for Remote Ground Sense	Installed
	5-6	Install to Disable This Offset/Remote Ground Sense Feature	Installed
E14	1-2	Jumper On - Channel 0 Bipolar Output Range Jumper Off - Channel 0 Unipolar Output Range	Installed
E15	Channel 1 Output Offset Select		
	1-2	NOT USED	Omitted
	3-4	Install to Use J1 BNC's Ground for Remote Ground Sense	Installed
	5-6	Install to Disable This Remote Ground Sense Feature	Installed
E16	1-2	Jumper On - Channel 1 Bipolar Output Range Jumper Off - Channel 1 Unipolar Output Range	Installed
E25	ADC Clock Source		
	1-2	DSP	Omitted
	2-3	Serial	Installed
E26 See Note 3	1-2 or 1-2 and 3-4	Supervisory and Nonprivileged	Installed
	3-4 Only	Nonprivileged	Installed
	None	Supervisory	
E27 See Note 3	1-2	Extended Addressing	Installed
	1-2 and 3-4 or None See Note 3	Standard Addressing	Omitted
E28	DSP Status Register Bit D12, Undefined Function		
	1-2	Jumper On - Logic 0 Jumper Off - Logic 1	Omitted
E29 See Note 3	DSP Status Register Bit D11, Factory Only		
	1-2	Jumper On - Logic 0 Jumper Off - Logic 1	Omitted

Note 1 - This offset is electronically limited to ±5.6VDC limits due to headroom of the output OP Amps.

Note 2 - If the user wishes the output connectors J0-J3 outer ground to be tied to the VME-4145's local analog ground, the user may install jumpers on both 3-4 and 5-6 pins.

Note 3 - The mezzanine board must be removed to modify this option. The options are located on the main board. It is advisable to have someone skilled in electronics remove this option board to prevent damage to the unit. Carefully grip firmly on all four corners of the mezzanine board to remove it from P8 and P9. Make sure all pins of the mezzanine's P1 and P2 are aligned correctly before reinstalling the unit.

Table 1-2 Jumper Description for Mezzanine Board

E17 See Note 4	Channel 2 Output Offset Select		
	1-2	Install for Channel 3 DAC to Offset DC Zero Point of Channel 2 Output. **DO NOT INSTALL 3-4 OR 5-6 WITH THIS OPTION, IT WILL SHORT OUT THE CHANNEL 3 DAC OUTPUT!**	Omitted
	3-4	Install to Use J2 BNC's Ground for Remote Ground Sense	Installed
	5-6	Install to Disable This Offset/Remote Ground Sense Feature	Installed
E18 See Note 4	1-2	Jumper On - Channel 2 Bipolar Output Range Select Jumper Off - Unipolar Output Range Select	Installed
E19 See Note 4	Channel 3 Output Offset Select		
	1-2	Not Used	Omitted
	3-4	Install to Use J3 BNC's Ground for Remote Ground Sense	Installed
	5-6	Install to Disable This Remote Ground Sense Feature	Installed
E14 See Note 4	1-2	Jumper On - Channel 3 Bipolar Output Range Select Jumper Off - Unipolar Output Range Select	Installed

Note 4 - The mezzanine board must be removed to modify this option. The options are located on the mezzanine board. It is advisable to have someone skilled in electronics remove this option board to prevent damage to the unit. Carefully grip firmly on all four corners of the mezzanine board to remove it from P8 and P9. Make sure all pins of the mezzanine's PL2 and P2 are aligned correctly before reinstalling the unit. The user should only have to access these option headers once to set up the user's initial application requirements.

Figure 1-2 Jumper Locations and Test Points on the Two-Channel Mezzanine Board

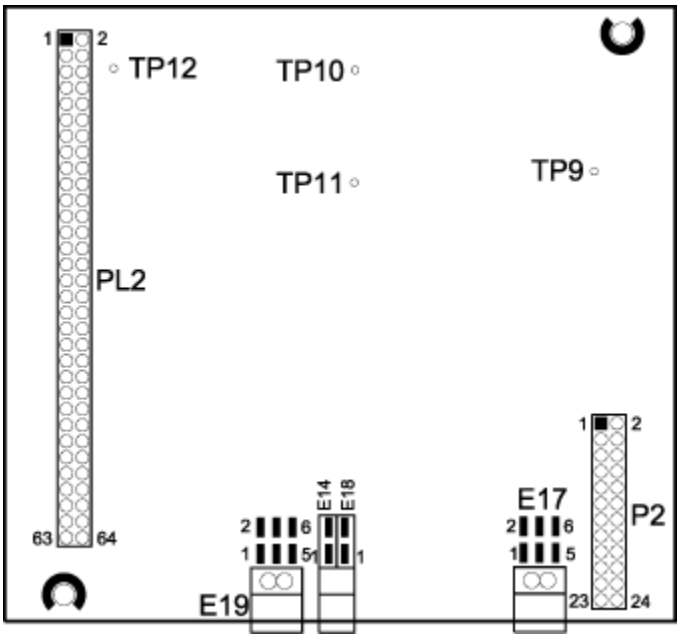
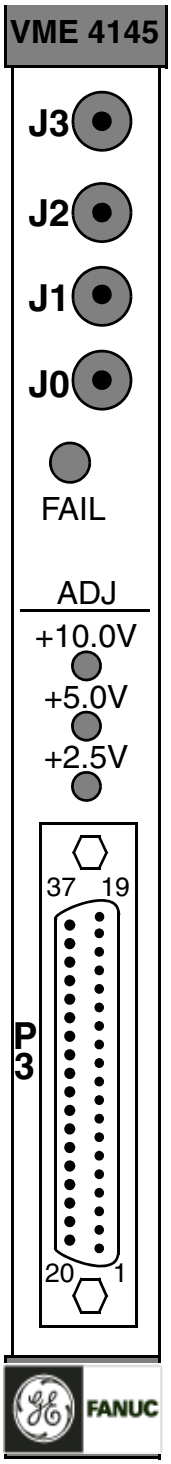


Figure 1-3 VME-4145 Front Panel



Front Panel Connector Pinouts

P3 PIN	ASSIGNMENT	I/O	P3 PIN	ASSIGNMENT
20	DIG GND	OUTPUT	1	FPSYNC0
21	DIG GND	OUTPUT	2	FPSYNC1
22	DIG GND	OUTPUT	3	FPSYNC2
23	DIG GND	OUTPUT	4	FPSYNC3
24	DIG GND	OUTPUT	5	FPCLK0
25	DIG GND	OUTPUT	6	FPCLK1
26	DIG GND	OUTPUT	7	FPCLK2
27	DIG GND	OUTPUT	8	FPCLK3
28	DIG GND	INPUT	9	FPTRIG0
29	DIG GND	INPUT	10	FPTRIG1
30	DIG GND	INPUT	11	FPTRIG2
31	DIG GND	INPUT	12	FPTRIG3
32	DIG GND	INPUT	13	FPCLKIN0
33	DIG GND	INPUT	14	FPCLKIN1
34	DIG GND	INPUT	15	FPCLKIN2
35	DIG GND	INPUT	16	FPCLKIN3
36	ALG GND	OUTPUT	17	VREF25
37	ALG GND	OUTPUT	18	VREF5
		OUTPUT	19	VREF10

J0 = CHN0

J1 = CHN1

J2 = CHN2

J3 = CHN3

Note: For two channel board J2 and J3 are dummy connectors.

1.5.1 VME Access

Figure 1-4 shows the Memory Mode Configuration while Figure 1-5 shows the VME access configurations supported by the VME-4145.

Figure 1-4 VME Memory Mode Configuration

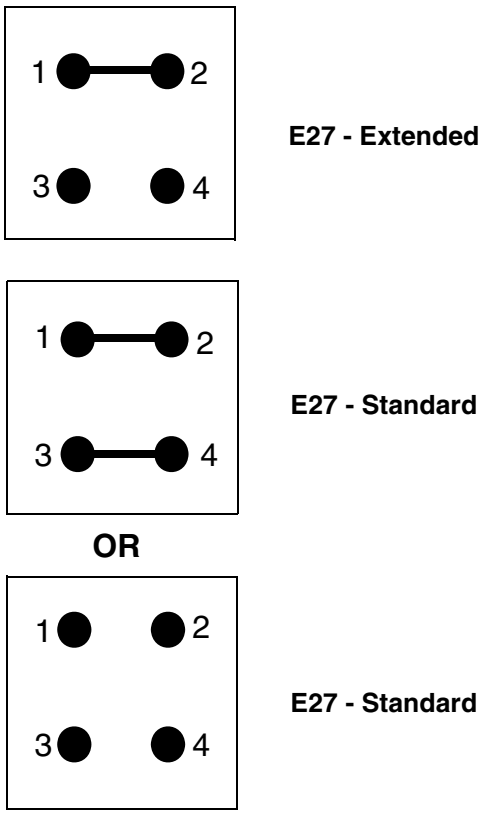


Figure 1-5 VME Access Mode Configuration

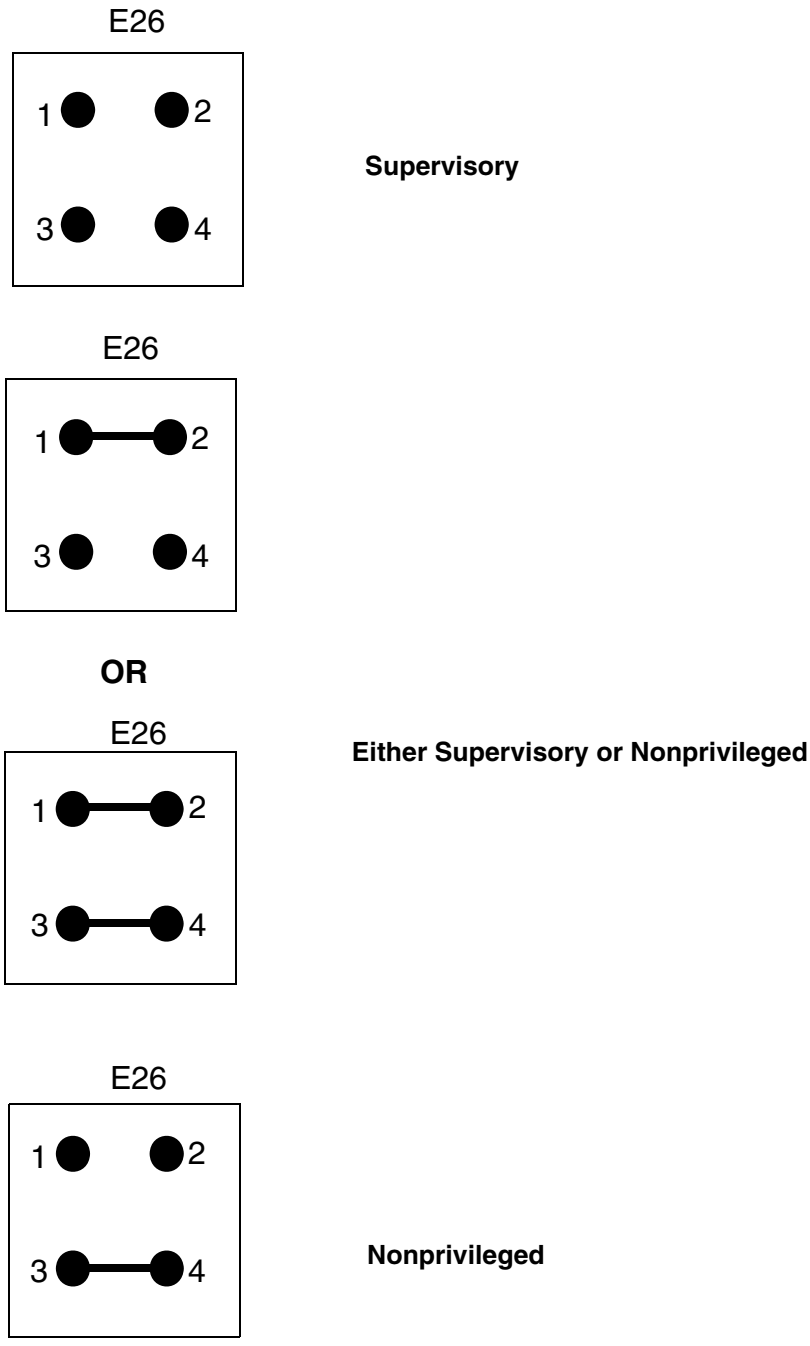
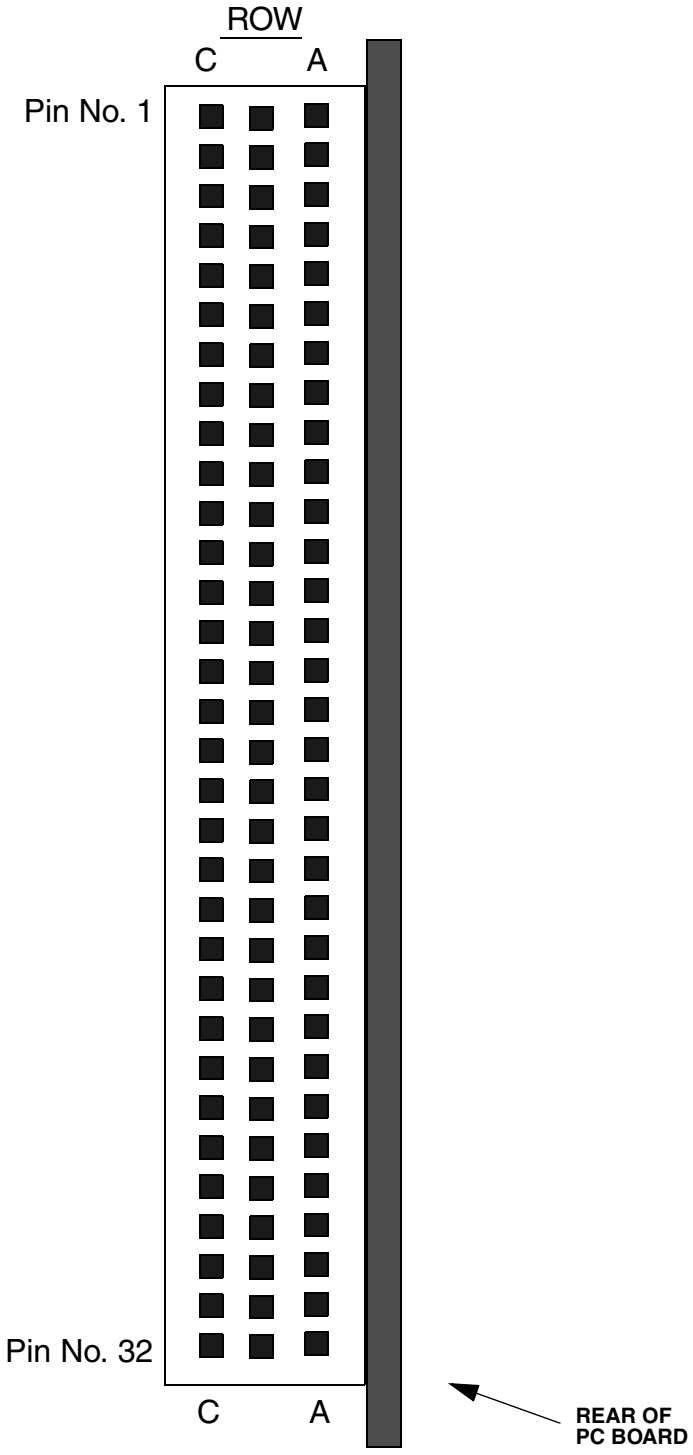


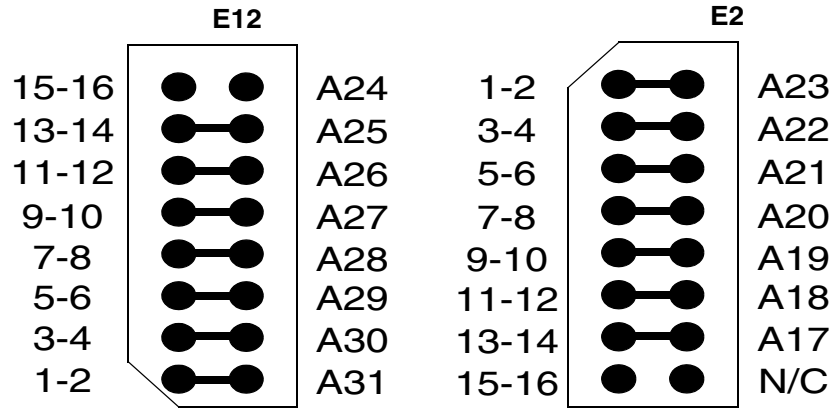
Figure 1-6 Pin Configuration for VME Interfaces P1 and P2



1.5.2 Base Address Jumpers

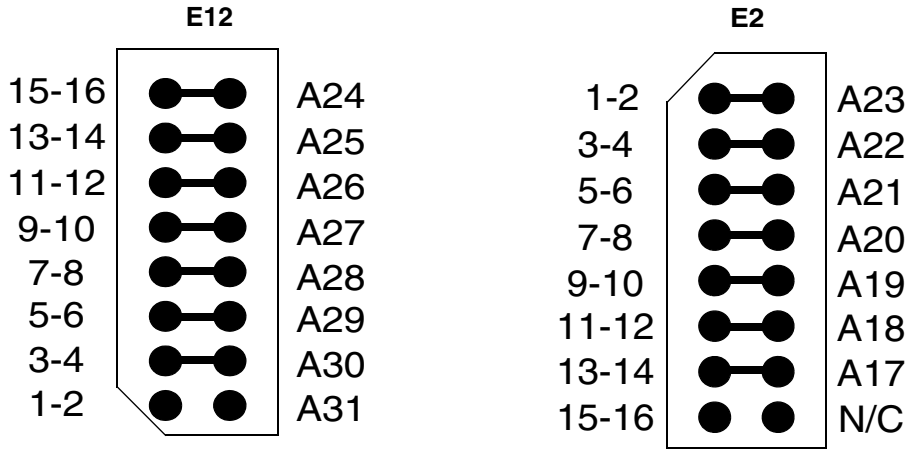
This set of jumpers is used to manually set the base address for the VME-4145 board. A jumper installed represents a logic 0, while a noninstalled jumper represents a logic 1. Figure 1-7 depicts the jumpers, E2 and E12, that are used to set the base address. Figure 1-7 and Figure 1-8 shows an example of how the base address jumpers represent the base address. In this example, the first 16 bits are hardwired to a logic 0.

Figure 1-7 Base Address Jumper Configuration



Example1: Base Address of 01000000

Figure 1-8 Example of Base Address Jumper Usage



Example 2: This Jumper Configuration Gives a Base Address of 0x80000000 when E26 and E27 are properly configured.

1.5.3 Two-Channel Mezzanine Board

You can install an optional two-channel mezzanine board, Figure 1-2. This board contains the channel 2 and 3 DACs, engines, and waveform RAMs. The output of this circuitry is fed back down to the main board where the deglitcher, filters, and output buffer circuitry completes the output path. Install the two-channel optional mezzanine board onto P8 and P9 on the main board. See Figure 1-1 for the location of these connectors. Refer to Figure 1-2 for the usage of the jumpers located on the mezzanine board.

1.5.4 Test Points

Table 1-3 Test Point Descriptions

Test Point	Description
TP1	+5 VDC Voltage Reference
TP2	+2.5 VDC Voltage Reference
TP3	DSP Flag Out Pin (FACTORY USE ONLY)
TP4	+10 VDC Voltage Reference
TP5	Channel 0 END_ADDRESS* (FACTORY USE ONLY)
TP6	Channel 0 Last Cycle* (FACTORY USE ONLY)
TP7	Channel 1 END_ADDRESS* (FACTORY USE ONLY)
TP8	Channel 1 Last Cycle* (FACTORY USE ONLY)
Mezzanine Test Points	
TP9	Channel 2 END_ADDRESS* (FACTORY USE ONLY)
TP10	Channel 2 Last Cycle* (FACTORY USE ONLY)
TP11	Channel 3 END_ADDRESS* (FACTORY USE ONLY)
TP12	Channel 3 Last Cycle* (FACTORY USE ONLY)

1.5.5 Potentiometers

Potentiometers R13, R11, and R9 are located on the front panel of the VME-4145 board. The description of these potentiometers is given below in Table 1-4.

Table 1-4 Potentiometer Descriptions

Potentiometer	Description
R13	+2.5 VDC Voltage Reference Adjust
R11	+5 VDC Voltage Reference Adjust
R9	+10 VDC Voltage Reference Adjust

1.5.6 Fuses

The VME-4145 uses one 10A axial wire lead fuse for the protection of the input of the DC-to-DC Converter.

1.5.7 Headers

The board contains 0.1 inch headers. These are industry standard with numerous suppliers.

1.6 Calibration

For maximum specified accuracy over time, the user must guarantee that the onboard voltage references remain precise. These references are used by the analog hardware and the DSP to calculate the gain and offset correction coefficients which are applied to the user's raw waveform data. Three precision references are used on the board: +2.5, +5, and +10 VDC.

1.6.1 Test Equipment Required:

- High accuracy and resolution Digital Voltmeter such as HP 3458A. The meter should display at least 6-1/2 digits or better and have a current NBS calibration.
- A small flat bladed screw driver or tuning tool (such as Hewlett-Packard Oscilloscope Probe tuning tool or jeweler's screw driver).

NOTE

It is highly recommended that the test points on the board be used when monitoring and adjusting the potentiometers for reference voltages.

1.6.2 +10.0 VDC Reference Voltage Calibration

Calibration Procedure:

- Monitor TP4 if you have a board extender to allow access to the side of the board.
- Analog ground return is at the GT1.
- If adjusting from the front panel, monitor the DB37, P3 pin 19.
- Analog ground return is DB37, P3 pin 37.
- Adjust front panel potentiometer R9 for +10.0000 VDC voltage reference.

NOTE

The +5 V reference is dependent on the +10 V reference. Adjust +10 V first, then the +5 V. Check both test points (tp4 and tp1) after making the adjustments to ensure that the voltages are accurate.

1.6.3 +5.0 VDC Reference Voltage Calibration

Calibration Procedure:

- Monitor TP1 if you have a board extender to allow access to the side of the board.
- Analog ground return is at the GT1.
- If adjusting from the front panel, monitor the DB37, P3 pin 18.
- Analog ground return is DB37, P3 pin 37.
- Adjust front panel potentiometer R11 for +5.0000VDC voltage reference.

1.6.4 +2.5 VDC Reference Voltage Calibration

Calibration Procedure:

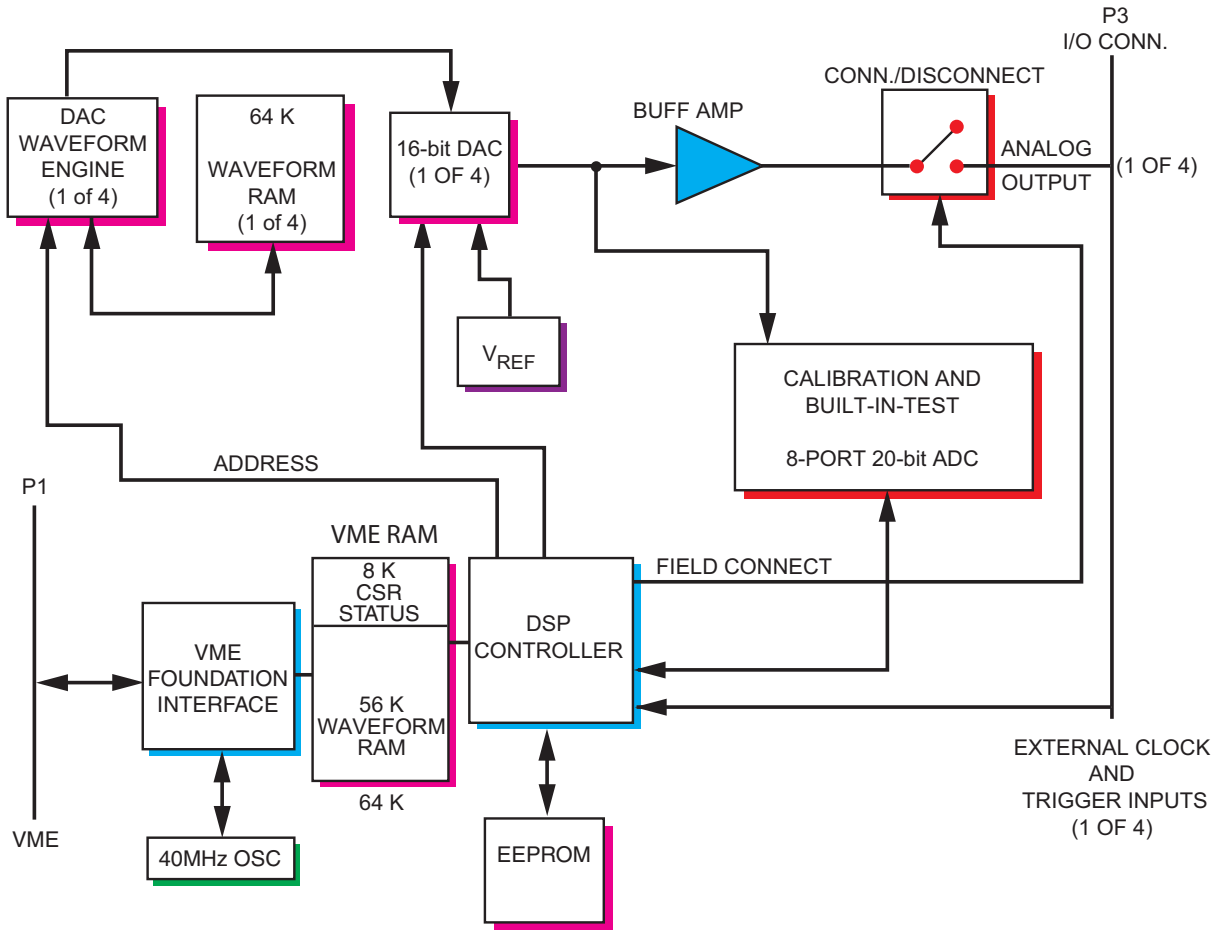
- Monitor TP2 if you have a board extender to allow access to the side of the board.
- Analog ground return is at the GT1.
- If adjusting from the front panel, monitor the DB37, P3 pin 17.
- Analog ground return is DB37, P3 pin 37.
- Adjust front panel potentiometer R13 for +2.50000VDC voltage reference.

2 • Theory of Operation

The VME-4145 provides four independent output channels. This capability is attained by the principal hardware functions, as shown in Figure 2-1 below. The VME-4145 contains the following principal functional elements:

- VME interface
- DSP
- EPROM/E2PROM
- DSP RAM and waveform RAM
- Control logic
- Timing and synchronization
- Analog output DACs
- Analog input ADC

Figure 2-1 VME-4145 Functional Block Diagram



2.1 General Operation

The following sections describe the process used to set up and operate the VME-4145.

There are three programmable operating modes available to the user:

- Continuous Mode Types I and II
- Single Cycle
- Continuous Burst

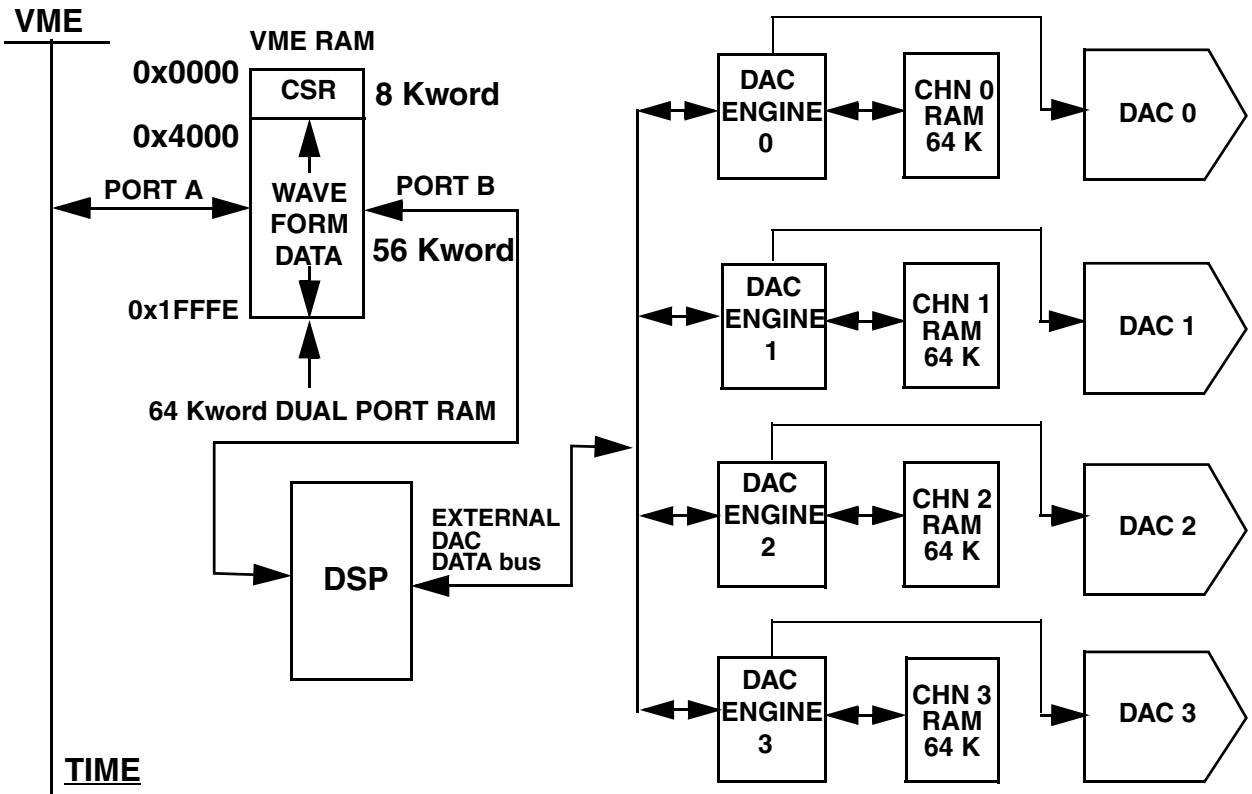
The sample clock source (internal or external) and period must also be programmed. The user can declare the use of an external trigger. The internal sample clocks are available at the front panel connector (P3). If the user has selected the Continuous Burst mode, the user must also set the idle time from one burst to the next burst. Both the sample clock and the buffer size can be programmed independently. Having a programmable buffer length allows the user the freedom from having to "fit" an integral number of waveform cycles in a fixed buffer size.

2.1.1 VME-4145 Onboard RAM

There are four RAM memories and one Common VME Waveform Data RAM memory associated with the VME-4145 board. Each of the four channels has its own dedicated 64K x 16-bit of RAM. The user can load the waveform table into the Common VME Waveform Data RAM in 56 Kword of address space, beginning at address 0x4000 through 0x1FFFE. The other 8 Kword of RAM is used by the CSR. Figure 2-2 is an illustration of the VME-4145 RAM structure.

For each channel there is a dedicated DAC. The DAC receives its output data from a separate channel RAM (one per channel). The user sets the size of the input buffer and downloads the desired waveform data into the desired Waveform Data RAM (the upper 56 Kword). The DSP will manipulate this data, applying calibration corrections. During calibration, a table of offset and gain coefficients is compiled and stored in RAM. There is an entry for offset and gain corresponding to each of the four channels configured in each of the five output voltage ranges.

Figure 2-2 VME-4145 RAM Structure



TIME

T0 PORT A = USER CONTROLS AND RETRIEVES STATUS BY WAY OF 8 Kword CSR. USER LOADS UP TO 56 Kword OF A WAVEFORM TABLE.

T1 PORT B = DSP RETRIEVES 56 Kword WAVEFORM. AUTOCALIBRATION PERFORMED IF REQUESTED.

T2 DSP MOVES UP TO 56 Kword OF WAVEFORM DATA TO ONE OF THE FOUR SEPARATE RAMS. THE DSP MUST WRITE TO THE DAC ENGINE OVER THE EXTERNAL DAC DATA BUS. THE DAC ENGINE WILL IN TURN WRITE THE DATA INTO ITS DEDICATED RAM.

T2 THE DAC ENGINE STEPS THROUGH THE WAVEFORM RAM TABLE ONE DATA POINT AT A TIME. THE DATA IS FED TO A DIGITAL-TO-ANALOG CONVERTER. THE RESULTING OUTPUT IS THE USER'S DESIRED ANALOG WAVEFORM.

2.1.2 Output Range

The VME-4145 generates bipolar and unipolar signals in the ranges stated in Table 2-1. The ranges are set up independently for each channel in the Channel Configuration register. Bipolar or unipolar signal outputs are jumper selectable.

Table 2-1 Bipolar and Unipolar Voltage Ranges

BIPOLAR and UNIPOLAR RANGES	
BIPOLAR	2.5VDC
	5VDC
	10VDC
UNIPOLAR	5VDC
	10VDC

NOTE

Unipolar 0 to 2.5V is not supported.

2.1.3 Data Engine Registers

Each channel has its own dedicated RAM and hardware control logic implemented in a large Erasable Programmable Logic Device (EPLD) that is called the "Data Engine,"refer to Figure 2-1 on page 29. The Data Engine accesses the RAM to supply samples as needed to the DAC input register.

2.1.4 Loading the Waveform Data Table

Waveform data is entered through a 64K x 16-bit waveform sample buffer. The buffer size is programmable from 2 to 65,536 samples per channel. The user configures a channel for either Continuous, Single Burst, or Continuous Types I and II Mode. The local DSP applies gain and offset corrections to the waveform samples, and stores them in the Channel RAM (not accessible by VME). Waveform generation begins with either a software trigger or an external trigger input. Each channel can use either an internal or external clock timebase input. During waveform generation, the trigger input is retriggerable and will restart an active waveform to the first location of the waveform table.

Each DAC, one per channel, has a 64K x 16-bit RAM Buffer. These buffers are not directly accessible to the user. The user must load the waveform table into the common Waveform Data RAM. The DSP will apply calibration offset and gain corrections to the data and transfer it to appropriate DAC engine's Channel RAM. The DSP accesses the Channel RAM through the DAC engine. Due to the Control and Status Register's overhead, the entire Common VME Waveform Data RAM is not available to the user for waveform downloading (8 Kword for the CSR and 56 Kword for the waveform data). Segmentation memory is allocated 2 Kword, starting at location 0x1000. If the user wishes to use segmented waveforms, the user must use a fixed starting address for that particular channel. Refer to Chapter 3, Programming. The download waveform data memory is allocated 56 Kword of memory space. For example, if the user requires the full 64 Kword for the waveform, they could load it in two parts: first, the 0 to 32 Kword part is loaded; and second, the load waveform command is issued. The user must poll the DSP for a command acknowledgment. This indicates the DSP has applied corrections and transferred the corrected data to the output buffer. The user then loads the 32 to 64 Kword portion and issues the second load memory command. The DSP will move Waveform Data RAM to DAC Engine Channel RAM based on

the VME start and end address pointers and DAC destination starting address pointers.

NOTE

The channel's range and data format must be set before loading a waveform for that channel.

2.2 Modes of Operation

The three user-programmable operating modes are: Continuous Type I or II, Single Cycle, or Continuous Burst modes. These modes are described in the following sections and in the programming section of this manual.

2.2.1 Continuous Modes Types I and II

In the Continuous Mode (Type I) of operation, the board will continually scan the waveform table and generate the desired signal until the board is halted. Therefore, once the waveform table reaches the end address, it resets the RAM address back to the first value and starts scanning through the data. In the Continuous Mode (Type II), the board scans the waveform table for a programmed number of (up to 510) cycles and then halts.

2.2.2 Single Cycle Mode

In the Single Cycle mode of operation, the board scans through the waveform table and generates a waveform until the end address is reached. At that point, the waveform halts. The voltage of the last waveform sample is maintained at the output. If so desired, the user can set the last value to 0x0000 for unipolar or 0x8000 for bipolar signals. This will leave the output at 0.00 VDC.

2.2.3 Continuous Burst Mode

In the Continuous Burst mode of operation, the board scans through the waveform table and generates a waveform until the $\mathbf{n}^{th}$ value in the table is reached. Then the waveform stops. The voltage of the last waveform sample is maintained at the output. At this time, a secondary counter with a value 'IDLE Timer' (the value 'IDLE Timer' determines the number of master clock cycles to wait) is decremented toward zero. This secondary counter is decremented at the master clock rate of 40 MHz or 25 ns ticks. When the counter reaches zero, the RAM address counter resets the address back to the starting address value and resteps through the data.

2.3 Idle Timer

The IDLE Timer is used to place idle time between each time the board scans the waveform table. Once the output waveform table has reached its $\mathbf{n}^{th}$ value, a count down IDLE Delay Timer is activated. When the idle counter reaches zero, the address counter is once again reset to the first value in the waveform table and the waveform is output again.

The value of 'IDLE Timer' must be set by the user. The default value is zero.

The range of 'IDLE Timer' is: $0 \leq M \leq 2^{24}-1$ (25 ns Ticks).

NOTE

If the value of 'IDLE Timer' = 0 is selected, the waveform is then generated repetitively with no delay.

2.4 Cycle Counter

The Cycle Counter (P) determines the number of waveform cycles to generate. Once the waveform table has reached its waveform sample value, a cycle counter is decremented. Once this cycle counter has reached zero, waveform generation is halted for that channel. The cycle counter may be reprogrammed while a waveform is running. If the count was set to 511, the waveform would be running continually. If the user then sets the count to one, the waveform would run through the waveform table for the last time and then stop. This is one way to **halt** the waveform and allow it to be retriggered at a later time. Do not set the count to zero while a waveform is running; the hardware will not operate correctly.

The value of the 'Cycle Counter' must be set by the user. The default value is 511 cycles.

The programming range of the 'P' is: 1≤ P ≤ 511 (0x1FF), with 'P' representing the Cycle Counter.

NOTE

If the value of P = 511 is selected, then an infinite number of cycles are generated. This is used to select the continuous mode (Type I) and continuous burst modes (Type II).

2.4.1 Examples of Setting Up Modes

Table 2-2 and Figure 2-3 on page 34 provides examples of different ways the user can set up the board to operate in one of the three modes using the IDLE Timer and Cycle Counter registers.

Figure 2-3 Example of the Burst Mode Type I

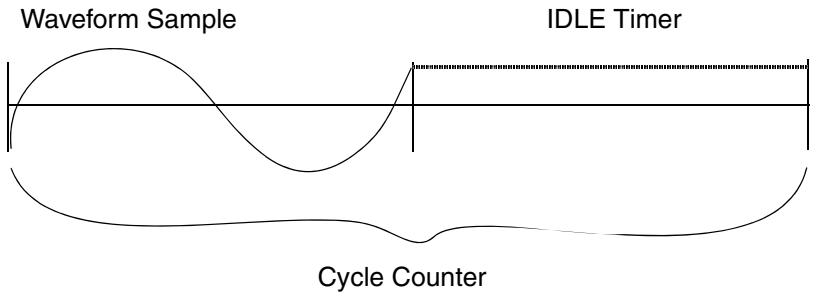


Table 2-2 Example Settings: Modes

Mode	Description	Waveform Sample Counter	IDLE Timer	Cycle Counter
Type I	Continuous	>2	0	511
Type II	Run for N cycles and stop	>2	0	254
Single Cycle	One burst and stop	>2	0	1
Burst IDLE Type I	Burst then wait 1,000 clocks for 5 cycles and then stop	>2	1,000	5
Burst IDLE Type II	Burst then wait 1,000 clocks, then repeated indefinitely	>2	1,000	511

2.5 Sample Clock and Trigger

The Sample Clock (on its rising edge) causes a read of the next output data value from the Waveform Data RAM and outputs to the DAC. The sample clock source (internal or external) and period must be programmed by the user. Each channel has its own dedicated clock source. This dedicated clock source allows the generation of four waveforms at different frequencies. Figure 2-4 on page 36 shows the sample clock and waveform output relationship. The trigger input is used to start a waveform output. The user may select to enable the external trigger function or use an internal software trigger.

The internal sample clock is a free running continuous clock. The clock can also be controlled from software and it can be programmed to a static logic one or zero. In this way, the software can generate ultralong period waveforms. Finally, each channel may select a common clock allowing two or more channels to have synchronized outputs. Channel zero's sample clock is the source of the common clock. The trigger input can be used to synchronize the output waveform with some external event. Please note that in Figure 2-4, the example shows the user has selected the external trigger mode. The waveform will not begin until the trigger has occurred.

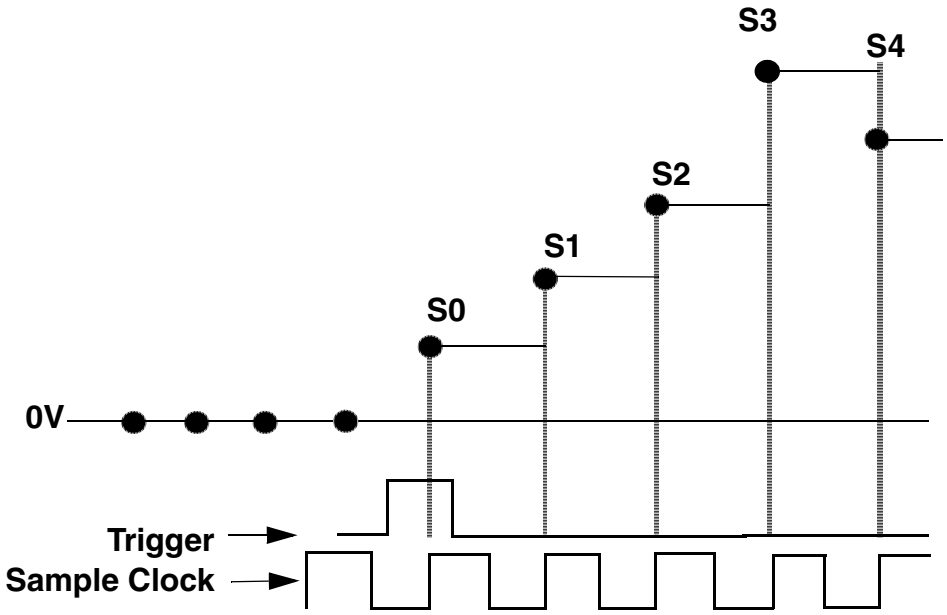
2.5.1 Internal Sample Clock

The maximum frequency of operation for the DACs is 2.5 MHz or 2.5 Million samples per second (Msps). This corresponds to a minimum period of 400ns per sample. The incremental step size is 25 ns from a 40 MHz master oscillator. The 20-bit Sample Rate registers (SR3-SR0) are set for the number of 25 ns clock cycles to wait per sample output. Both the sample clock and the buffer size can be programmed independently. Having a programmable buffer length frees the user from having to 'fit' an integral number of waveform cycles in a fixed buffer size. The sample clocks may be changed while a waveform is running.

Internal Sample Clock Specification:

400 ns to 26.6 ms (with 25 ns resolution).
(SR3-SR0) = 0X00010 to 0XFFFFF

Figure 2-4 Sample Clock



Trigger: Starts burst waveform generation
Sample Clock: New data sample every rising edge of clock (after trigger)
S0, S1, S2, S3, S4,.....Sn-1, Sn: Waveform Samples. Programmed using WSB.
Ts: Sample period. Programmed using SRx.

2.5.2 External Sample Clock

Each channel has an external input for a user-supplied sample clock. If the user decides on using the external clock instead of the internal clock, the external clock must be guaranteed within its clock specifications of voltage, period, and duty cycle. The programming is discussed in Chapter 3, Programming. Also refer to, Front Panel Signal Buffers, for a detailed explanation of the front panel signal buffers.

External Sample Clock Specification:

- **Period:** 400 ns period (minimum)
- **Clock Pulse Width:** >120 ns
- **Voltage:** CMOS TTL compatible (Low <0.8 VDC, High >3.15 VDC)

NOTE
It is recommended that the user use the internal clock while setting up the engine and downloading a waveform. Once this is done, the user can switch to the external clock to output the waveform.

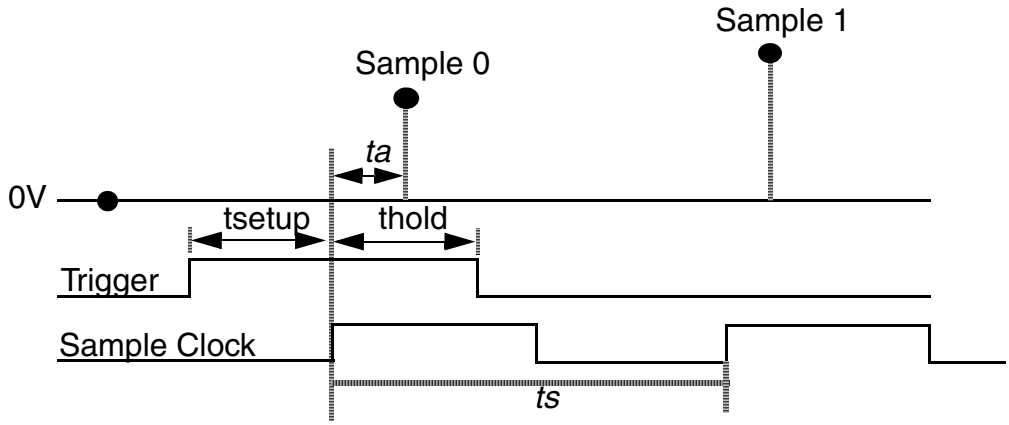
2.5.3 External Trigger

Each channel has an external input for a user-supplied trigger. If the user decides on using an external trigger instead of a software trigger, the user must guarantee this trigger is within the specifications of voltage and period. It must also meet the setup and hold times specified. See Figure 2-5 for details.

- **Voltage:** CMOS TTL Compatible (Low <0.8 VDC, High >3.15 VDC)
- **tp Trigger Period:** 500 ns period (minimum)
- **ts Setup Time:** >50 ns

- **th Hold Time:** >100 ns
- **Trigger Latency:** 300 ns
- **ta Clock edge to Analog value change:** 200 ns (maximum)

Figure 2-5 External Trigger



- **tsetup:** Setup time before rising edge of sample clock.
- **thold:** Hold time after rising edge of sample clock.
- **tp:** Minimum pulse width for trigger pulse.
- **Sample Clock:** New data sample every rising edge of clock (after trigger).
- **S0,S1,...Sn-1, Sn:** Waveform Samples. Programmed using WSB.
- **Ts:** Sample period. Programmed using SRx.

2.6 Hardware States

The channel's control hardware has six operating states: INIT, LOAD, ARM, WAVE, IDLE, and DONE states.

In the ***LOAD*** state, the board is ready to load a waveform. The user loads the waveform table, and can elect to take all channels offline. The user also elects to leave the DAC at the last table value it generated.

In the ***ARM*** state, the board is waiting for a software trigger or an externally supplied trigger. Once the board is "triggered," the state becomes WAVE and the first waveform RAM buffer value is read and sent to the DAC.

In the ***WAVE*** state, the board is actively reading from the RAM and writing to the DAC at the sample clock frequency. When P = 511, the WAVE state is never exited until a user software HALT command is given. When P<511, the board will run until the desired number of waveform cycles has been generated. Then the board enters either the ARM or DONE state.

In the ***IDLE*** state, the board is holding at the last value in the table.

In the ***DONE*** state, the board will automatically enter the ARM state. The **DONE** state is transitioned through faster than the DSP can capture. Therefore, the user should not attempt to monitor for this state. Once a waveform halts, it will automatically rearm and wait for a new trigger.

The user enters a particular channel into the DONE state by issuing the channel HALT command. This leaves the DAC output value at the last value written.

2.7 VME Interface

The VME-4145 responds to word (D16) or byte (D08) data accesses. Supervisory, nonprivileged, or both access modes are supported along with extended and standard address modes.

2.8 Commands

The user can issue a number of commands to the VME-4145 by way of the VME. The commands are initiated by the VME byte write access to the indicated address. The section in Chapter 3 titled *VME-4145 Commands* discusses the various commands that can be issued through the Control Registers along with the required parameters and the expected responses.

2.9 Calibration

The initiate calibration command causes the board to stop processing data, enter the calibration mode, and blink the front panel LED. Any VME read accesses during the calibration period will return a value of 0x3C00. All VME writes will be ignored.

The DSP utilizes the 20-bit Sigma-Delta ADC to read each of the output channels at various output values from 0x0000 to 0xFFFF. First, the ADC is self-calibrated. The ADC's programmable attenuator is then analyzed for gain and offset errors using the onboard 2.5, 5, and 10 V precision references. Gain and offset correction factors are then calculated. Finally, each DAC channel is analyzed for all ranges supported by the current unipolar/bipolar hardware strap.

Once the DSP recognizes a user calibration request, there are a number of calculations the DSP will process. Therefore, during this time interval, the VME-4145 will be busy and not able to generate any waveforms or respond to other commands. A large number of samples are taken for each ADC reading. These multiple readings per data point are averaged to reduce noise-related errors. Finally, all of the data points are analyzed and a "Least Square Statistical Algorithm" is applied to determine the gain and offset coefficients. Typically, the DSP will be busy for over 10 to 20 minutes of processing time.

When autocalibration is initiated by a software command, an embedded DSP loads calibration values into each of the output DACs. The output voltages are read back into the DSP through a 20-bit Sigma-Delta ADC. This is repeated until a sufficient number of calibration points have been measured. A calibration table consisting of offset and gain corrections for each of the four outputs in each of the voltage ranges is compiled and stored in RAM. These correction factors are used to correct the data when the waveform sample buffer is loaded.

NOTE

Due to the lengthy calibration time, the unit is not automatically calibrated at each power on cycle of the board. However, once the board has been calibrated, the stored coefficients may be recalled for reuse.

As a first step, the ADC is given a self-calibrate command. The 20-bit ADC will autocalibrate itself by grounding its own input and reading the external voltage reference.

After the ADC is calibrated, the programmable attenuator in front of the ADC is analyzed to determine its offset and gain errors using the onboard precision 2.5, 5, and 10 VDC references.

NOTE

It is important that the user guarantee the accuracy of these references since they are the basis of all calibration for the board.

Now that the ADC and its input attenuator are both calibrated, the DSP can analyze each of the DAC channels to determine their independent gain and offset coefficients.

Please note that the unit may only calculate DAC coefficients depending on how each channel is configured prior to the calibration request. Due to a manual user hardware strap option that determines whether the DAC is a bipolar or unipolar output, the DSP will only calibrate for that configuration. For example, if the user has a channel set for unipolar output, only the 0 to +10 and 0 to +5 VDC coefficients will be determined. Likewise, if the user has a channel set for bipolar output, only the ±2.5, ±5, and ±10 VDC coefficients will be determined. If the user changes this hardware strap, the user should recalibrate the board for the new ranges. The DSP cannot read these unipolar/bipolar straps directly. The user must inform the DSP through the Channel Configuration Registers (CCRs) using Bit 12. It is important that the user set all four of the CCRs prior to invoking calibration. This will inform the DSP which coefficients it should calculate (bipolar or unipolar).

After calibration has completed, the new calibration coefficients will remain in volatile RAM. They will remain usable until the board's power has been removed. If the user wishes to retain these values permanently in E^2PROM, the user must issue the upload RAM coefficients to E^2PROM command. The user may upload to E^2PROM 10,000 times. The data will be retained for at least ten years. If the 10,000 time limit is exceeded, the socketed PLCC E^2PROM device will need to be replaced.

The user has two options once the VME-4145 is powered up. With the first option, the user may issue a calibration command and wait for autocalibration to finish. This method is suggested for users requiring high accuracy DC values or where there are extreme temperature variations. This method will, of course, require the VME-4145 to remain busy for a period after powerup. The second option is to issue a download E^2PROM coefficients to RAM command. The previously determined coefficients are retrieved from permanent storage and moved into the working volatile RAM memory of the DSP. If the user is more interested in AC waveforms, then this method will be attractive since it is the faster of the two.

2.10 Digital Signal Processor (DSP)

The DSP is responsible for gathering calibration data from the ADC and providing offset and gain correction to the user's waveform data. The DSP and EPLD combine to control all board operations. A **least means square** approximation of the channel transfer function is determined using this data.

The gain and offset coefficient for each channel are calculated from this estimate and stored internally. When calibration is complete, the DSP sets a bit to inform the user of calibration completion.

After the user loads a new waveform, the DSP applies the gain and offset calibration coefficients. Calibration is performed sample-by-sample as it is moved from the VME RAM to DAC RAM.

2.11 EPROM

The DSP firmware is stored in the EPROM. There are four main boot pages stored in the EPROM: the power up/reset boot page, main routine boot page, calibration boot page, and the reconfiguration boot page.

2.12 E^2PROM

The gain and offset coefficients calculated in the Calibration mode are stored in the E^2PROM. The board is shipped with factory-determined bipolar coefficients. When the user initiates calibration, newly calculated coefficients are stored in the DSP for use in real-time correction. The new coefficients are not automatically stored in the E^2PROM. A command must be issued to upload the new coefficients to the E^2PROM for permanent storage. This allows the user to run new correction coefficients that are not stored in the E^2PROM. Most commercially available E^2PROMs have a guaranteed 10,000 write cycle lifetime. A E^2PROM Writes Register (EWR) is available to the user indicating how many writes have occurred. If this number gets close to 10,000 limit, it is suggested that the E^2PROM be replaced so data is not lost or corrupted. The E^2PROM also contains 12 Kword of data which configures the DAC Data Engines on power up. The user does not have direct access to this memory.

2.13 Control Logic

The control logic consists of the logic required to read and write the RAM and E^2PROM, as well as, load and initiate waveform generation and arbitration between the board and the VME. The control logic consists of an EPLD.

2.14 Analog Circuitry

The digital data from each waveform buffer is fed to a DAC. The range options are handled at this point as well. The range adjust output is fed to a track-and-hold deglitcher. The track-and-hold output is then routed to an optional low pass filter circuit. The output of the filter is routed to a high-current driver Op-Amp capable of driving high capacitance loads. An analog switch is included to disconnect the output from field wiring. This allows the board to test the DAC without affecting external devices. A 20-bit ADC is used to calculate calibration parameters for the DAC.

2.14.1 Digital-to-Analog Converter (DAC)

The DAC is a monolithic high-performance device which accepts 16 bits of digital data and converts it into an analog representation. The current output of the DAC is routed into a current to voltage conversion Op-Amp. The voltage output of the Op-Amp is fed into track-and-hold circuitry. This reduces output glitches which can occur during DAC conversions.

2.14.2 Analog-to-Digital Converter (ADC)

The ADC uses a Sigma-Delta conversion technique to digitize input signals with up to 20 bits of no missing code performance. There is one ADC on the board; its input utilizes an analog multiplexer to select one of the four channels. In front of the ADC is a programmable attenuator. This provides the nominal 2.5VDC magnitude input range of the ADC. The ADC is controlled by the DSP.

2.14.3 Analog Output Filtering (Optional)

Each channel can be routed through a four-pole Salen-Key filter. For applications requiring low pass filters, GE Fanuc Intelligent Platforms provides optional Filters assembled per channel. All filters are Fourth-Order, Salen-Key Multiple Feedback Low Pass Filters. Check Table 2-3 below for currently available filters and cutoff frequencies.

Table 2-3 Filters Available for the VME-4145

FILTER	TYPE	CUT OFF FREQUENCY
VME-4145-10x	Chebyshev	500 kHz
VME-4145-20x	Chebyshev	1.0 MHz
VME-4145-30x	Bessel	500 kHz
VME-4145-40x	Bessel	1.0 MHz
VME-4145-00x	None	(x = 0, 4 CH) (x = 1, 2 CH)

2.14.4 Self-Test

Self-Test is run automatically after a system reset or software command. The Self-Test Status Registers (SSRs) indicate the success or failure of each channel. Each channel is indicated by a separate bit in the register.

General failure (not channel specific) is indicated in the General Test Status (GTS) Register. Channel specific failures are reported in their corresponding Self-Test Status (STS CH 0-3) Registers.

2.14.5 System Reset

After a system reset, all outputs are in the offline mode; all control registers are in their default state; and Self-Test is initiated. The software reset register is set to 0x1D, indicating a reset has occurred.

2.14.6 Transfer Function

The output voltage generated is controlled by the digital value it is fed as well as the output range selections. The equation below describes the board's transfer function:

$$E_{OUT} = E_{OUTMIN} + \left\{ \frac{N_{DATA}}{65{,}536} xE_{SPAN} \right\}$$

Where:

E_{OUT} = Channel output voltage

E_{OUTMIN} = Negative end of the range

N_{DATA} = Channel data from the waveform data table (0x0000 to 0xFFFF)

E_{SPAN} = Positive end of the range minus the Negative end of the range

Example for the ±5V range:

Let N_{DATA} = 0x8000 of decimal 32768, midscale value

$$E_{OUT} = -5\ V + \left(0x8000 \quad X \quad \frac{+10\ V}{65536} \right)$$

$$= -5\ V + 0x8000\ X\ 0.000152 = -5\ V + 5\ V = 0\ V$$

2.14.7 Output Impedance and Drive

The board exhibits very low impedance in the online condition (typically less than 1.5 Ω) and very high impedance (typically more than 10M Ω) in the offline state. The board provides high-current drivers (±10 mA) for low impedance (minimum 1 kΩ) loads.

2.14.8 Output Short Circuit Protection

The outputs are protected from indefinite shorts to GND and from ±25 VDC transients for one second.

2.14.9 Field Disconnect

The board provides solid-state relays which can disconnect the VME-4145's four channels from any existing field wiring. This allows the board to perform full-range calibration test of the DAC without external devices being disturbed. A separate field disconnect control bit is used for each output channel.

2.14.10 Sync Pulse Output

When a channel is in the WAVE mode (active output), a one clock cycle wide pulse is generated each time the first location of the waveform is transmitted. This marker pulse may be used by other external devices to synchronize events. This

signal is a TTL-level compatible digital signal, active high or low (software selectable). It is capable of driving one CMOS or TTL-level load. The signal is available at the front panel connector (P3). The polarity is programmable in the Channel Configuration Register (CC0 - CC3).

2.14.11 Front Panel Status LED

The front panel indicator is illuminated after a system reset and is extinguished upon the successful completion of Self-Test. While executing commands such as Autocalibration or Built-in-Test (BIT), the LED will blink and extinguish upon successful completion. The LED can also be turned On or Off under software control.

2.14.12 Front Panel Reference Voltage Access

A front panel connector allows access for measuring the reference voltages. Located adjacent to and below the reference voltage connector is a front panel access port to the reference voltage adjustments. Autocalibration is based on these precision reference voltages. Three reference voltages are used on the board +10 V, +5 V, and +2.5 VDC. For precision calibration, it is recommended to use the ground tie point GTI and test points on the PCB. TP1 = +5, TP2 = 2.5, and TP4 = +10 VDC.

2.14.13 Front Panel Clock Connection

The user can supply an external sample clock on the P3 front panel connector. Each channel has a dedicated clock. This provides independent sample output timing for each channel. This signal must conform to TTL levels. The user can also monitor the system's internal sample clock for each channel. P3 has pins dedicated to these TTL-level clock outputs.

2.14.14 Front Panel Signal Buffers

All logic signals on the P3 connector are routed through a Schmidt Trigger 74HC14 buffer. The buffers have a typical propagation delay of 31 ns maximum at T_A = 25° C. High-level input V_{IN} minimum 3.15 V. Low-level input V_{IL} maximum 0.9 V.

3 • Programming

The VME-4145 contains many user-configurable registers. *"Getting Started"* gives some guidelines on how to get started using the VME-4145. *"VME Memory Map"* contains the overall memory map that shows all of the registers and their respective memory addresses that are used on the VME-4145 board. *"Register Definitions"* describes the general registers that are to be configured before the configuration commands are issued. Some of the registers contained in this section are general input and output registers for the configuration commands. *"VME-4145 Commands"* describes the configuration commands and the registers that are used by these commands.

3.1 Getting Started

There are four C code listings provided in ***"Appendix A: Sample Codes"*** to help in getting started. These were compiled under Cross Code C for the 68000. They should compile on other compilers with few modifications. The source code files are defined as follows:

- **4145.h** Common header file for the three C routines. This file defines the structures of all of the registers used by the VME-4145, the segmentation programming areas, and the VME waveform RAM.
- **user2.c** This code sets the board up to generate a simple square wave.
- **user3.c** This code sets the board up to generate a square/triangle segmented waveform.
- **user4.c** This code sets the board up to generate a sinusoidal waveform.

3.1.1 Sequence of Initialization

The order in which certain registers in the VME-4145 are initialized is extremely important. Examine the examples carefully and note the following suggested sequence. Suggested sequence for initializing the VME-4145:

1. Set up DAC Channel Pointer Register (0x0002) to select the desired channel.
2. Issue the Disable Segmentation command (0x001A) and wait for response.
3. Issue Initialize CHN Engine/Halt Waveform Generation command (0x0005) and wait for response.
4. Set desired parameters in the General Configuration Register (0x000A).
5. Issue Load General Configuration command (0x0001) and wait for response.
6. Load desired waveform data into VME waveform RAM.
7. Set waveform RAM starting and ending address registers.
8. Set VME RAM starting and ending address registers.
9. If waveform data is already loaded into the waveform RAM, issue Load Start/End Address pointers command (0x0013) and wait for the response.
10. Issue Load Waveform Command (0x0006) or (0x0023) and wait for response.
11. Set desired parameters in the appropriate Channel Configuration Register (0x0028 - 0x002E).
12. Issue Load Channel Configuration command (0x0003) and wait for response.

13. Set desired sample rate in the appropriate Sample Rate Registers (0x0030/ 0x0032 - 0x003C/0x003E).
14. Issue Load Sample Rate command (0x0009) and wait for response.
15. Set desired idle timer value in the appropriate Idle Timer Registers (0x0044/ 0x0046 - 0x004C/0x004E).
16. Issue Load Idle Timer command (0x0004) and wait for response.
17. Set desired cycle count in the appropriate Repeat Count Register (0x0068 - 0x006E).
18. Issue Load Repeat Count command (0x0012) and wait for response.
19. Set desired clock source in the Clock Mux Register (0x00C0).
20. Issue Load Clock Mux command (0x0014) and wait for response.
21. Issue Software Trigger CHN command (0x0007) and wait for response.
22. Observe Waveform at the desired output jack.

The above list is only a suggestion and does not have to be performed exactly as listed. A user's application may call for some deviations from this list. The following are three guidelines which **must** be followed:

1. The General Configuration Register (GCR) must be programmed before attempting to issue the load waveform command, since the GCR instructs the DSP which format the download data will take (two's complement or binary).
2. The channel configuration registers must be programmed.
3. The Repeat Count Register (RCR) must be programmed after a waveform has been downloaded. The act of writing a nonzero value to this register causes the engine to move from the **LOAD** state to the **ARMed, waiting for trigger** state.

3.1.2 Guidelines on Defining Waveforms

The analog circuitry contains many operational amplifiers from the actual DAC to the output connector. All op-amps exhibit a phenomenon called "rate" typically defined in units of volts per microsecond. This rate defines how fast an op-amp can change its output voltage from one value to another. Instead of an ideal straight line from one value to the next, a ramp is actually generated from one voltage to the next. An ideal op-amp would have no slew rate. Unfortunately, all real world op-amps have some slew rate to a varying degree.

The board also contains a track-and-hold circuit. During certain transitions from one value to another, DACs will generate "glitches" in its output. During this transitional window, the Track-and-Hold circuit freezes the old value by going into the **hold** mode. It then waits for the glitch window to pass. At this point, it goes back to the **track** mode and ramps up to the new stable output value. While the track-and-hold circuit does drastically reduce any glitches, it does add another delay in transitioning from one value to the next.

Slew rate was mentioned because when the waveform table is defined for downloading, slew rate affects the output of the waveform table. For instance, the following table defines a square wave:

- 0x0000
- 0xFFFF
- 0x0000
- 0xFFFF

This table would produce a square wave having multiple cycles. However, in the case of ±10 V bipolar outputs, this table is demanding instantaneous swings of 20 V from -10 V to +10 VDC. It would be preferable to define a square wave with the following format:

- 0x0000
- 0x0000
- 0x0000
- 0x0000
- 0xFFFF
- 0xFFFF
- 0xFFFF
- 0xFFFF

This would produce a square wave having one cycle in the waveform table. This only requires the hardware to ramp up from -10 V to +10 V once. The sample output rate can be increased to overcome the longer waveform period.

The two parameters that the user can control are the number of data points in the waveform table and the actual output rate for scanning the waveform table. The formula has been defined in a previous section.

The more precise or ideal the desired waveform needs to be, the more table values or samples that are required. This longer table may require an increase in the output rate. However, the faster the output rate and the wider the difference between adjacent samples, the more slew rate becomes a factor.

The smaller the waveform is in number of table samples, the faster a waveform may be generated (its frequency). In the case of a sinusoid, the lower the number of samples, the more stair-steps become apparent.

Unfortunately, these two parameters are interrelated and trade-offs must be made when programming a waveform. The user may wish to try a small table and slow clock and then a large table and fast clock to see which gives a more suitable situation. Avoiding adjacent samples that move from minimum range to maximum range, such as a square wave is another way of reducing the slew rate. One or more intermediate points between these would be preferable.

3.2 VME Memory Map

Table 3-1 VME Memory Map

OFFSET ADDRESS (HEX)	NAME	DESCRIPTION
0000	BID	BOARD ID
0002	CHN	DAC CHANNEL POINTER
0004	VCR	VME CMD REGISTER
0006	CDR	CMD DATA REGISTER
0008	CRR	CMD RESPONSE REGISTER
000A	GCR	GENERAL CONFIG REGISTER
000C	SSR	SELF-TEST STATUS REGISTER
0012	RST	SOFTWARE RESET
0014	EWR	EEPROM WRITE CYCLES REGISTER
0016	CMD_DATA_LSW	AUXILIARY DATA REGISTER LSW
0018	CMD_DATA_MSW	AUXILIARY DATA REGISTER MSW
001A	BIT_ADC_HI	4 MS bit's ADC VALUE
001C	BIT_ADC_LO	16 LS bit's ADC_VALUE
001E	FRR	FIRMWARE REVISION REGISTER
0020	CSR0	CHAN STATUS REGISTER 0
0022	CSR1	CHAN STATUS REGISTER 1
0024	CSR2	CHAN STATUS REGISTER 2
0026	CSR3	CHAN STATUS REGISTER 3
0028	CCR0	CHAN CONFIGURATION REGISTER 0
002A	CCR1	CHAN CONFIGURATION REGISTER 1
002C	CCR2	CHAN CONFIGURATION REGISTER 2
002E	CCR3	CHAN CONFIGURATION REGISTER 3
0030	SRH0	SAMPLE RATE HIGH 4 bits CHN 0
0032	SRL0	SAMPLE RATE LOW 16 bits CHN 0
0034	SRH1	SAMPLE RATE HIGH 4 bits CHN 1
0036	SRL1	SAMPLE RATE LOW 16 bits CHN 1
0038	SRH2	SAMPLE RATE HIGH 4 bits CHN 2
003A	SRL2	SAMPLE RATE LOW 16 bits CHN 2
003C	SRH3	SAMPLE RATE HIGH 4 bits CHN 3
003E	SRL3	SAMPLE RATE LOW 16 bits CHN 3
0040	IH0	CHN 0 IDLE HIGH WORD (8 bits)
0042	IL0	CHN 0 IDLE LOW WORD (16 bits)
0044	IH1	CHAN 1 IDLE HIGH WORD

Table 3-1 VME Memory Map (Continued)

OFFSET ADDRESS (HEX)	NAME	DESCRIPTION
0046	IL1	CHAN 1 IDLE LOW WORD
0048	IH2	CHAN 2 IDLE HIGH WORD
004A	IL2	CHAN 2 IDLE LOW WORD
004C	IH3	CHAN 3 IDLE HIGH WORD
004E	IL3	CHAN 3 IDLE LOW WORD
0050-0066		RESERVED
0068	RCR0	REPEAT COUNT REGISTER CHAN 0
006A	RCR1	REPEAT COUNT REGISTER CHAN 1
006C	RCR2	REPEAT COUNT REGISTER CHAN 2
006E	RCR3	REPEAT COUNT REGISTER CHAN 3
0070		CHANNEL AUTO DETECT
0071		FACTORY CODE
0072-007E		RESERVED
0080	SARH0	CHAN 0 START ADDRESS HIGH (1 bit, MSW)
0082	SARL0	CHAN 0 START ADDRESS LOW (16 bits, LSW)
0084	EARH0	CHAN 0 END ADDRESS HIGH (1 bit, MSW)
0086	EARL0	CHAN 0 END ADDRESS LOW (16 bits, LSW)
0088	DSRH0	CHAN 0 DESTINATION START ADDR (1 bit, MSW) HIGH
008A	DSRL0	CHAN 0 DESTINATION START ADDR (16 bit, LSW) LOW
008C	DERH0	CHAN 0 DESTINATION END ADDR (1 bit MSW) HIGH
008E	DERL0	CHAN 0 DESTINATION END ADDR (16 bit LSW) LOW
0090	SARH1	CHAN 1 START ADDRESS HIGH (1 bit, MSW)
0092	SARL1	CHAN 1 START ADDRESS LOW (16 bits, LSW)
0094	EARH1	CHAN 1 END ADDRESS HIGH (1 bit, MSW)
0096	EARL1	CHAN 1 END ADDRESS LOW (16 bits, LSW)
0098	DSRH1	CHAN 1 DESTINATION START ADDR (1 bit, MSW)
009A	DSRL1	CHAN 1 DESTINATION START ADDR (16 bit, LSW)
009C	DERH1	CHAN 1 DESTINATION END ADDR (1 bit MSW)
009E	DERL1	CHAN 1 DESTINATION END ADDR (16 bit LSW)
00A0	SARH2	CHAN 2 START ADDRESS HIGH (1 bit, MSW)
00A2	SARL2	CHAN 2 START ADDRESS LOW (16 bits, LSW)
00A4	EARH2	CHAN 2 END ADDRESS HIGH (1 bit, MSW)
00A6	EARL2	CHAN 2 END ADDRESS LOW (16 bits, LSW)
00A8	DSRH2	CHAN 2 DESTINATION START ADDR (1 bit, MSW) HIGH
00AA	DSRL2	CHAN 2 DESTINATION START ADDR (16 bit, LSW) LOW
00AC	DERH2	CHAN 2 DESTINATION END ADDR (1 bit, MSW) HIGH

Table 3-1 VME Memory Map (Continued)

OFFSET ADDRESS (HEX)	NAME	DESCRIPTION
00AE	DERL2	CHAN 2 DESTINATION END ADDR (16 bit, LSW) LOW
00B0	SARH3	CHAN 3 START ADDRESS HIGH (1 bit, MSW)
00B2	SARL3	CHAN 3 START ADDRESS LOW (16 bits, LSW)
00B4	EARH3	CHAN 3 END ADDRESS HIGH (1 bit, MSW)
00B6	EARL3	CHAN 3 END ADDRESS LOW (16 bits, LSW)
00B8	DSRH3	CHAN 3 DESTINATION START ADDR (1 bit, MSW) HIGH
00BA	DSRL3	CHAN 3 DESTINATION START ADDR (16 bit, LSW) LOW
00BC	DERH3	CHAN 3 DESTINATION END ADDR (1 bit, MSW) HIGH
00BE	DERL3	CHAN 3 DESTINATION END ADDR (16 bit, LSW) LOW
00C0	CMR	CLOCK MUX REGISTER
00C2	STS0	SELF-TEST STATUS CHN0
00C4	STS1	SELF-TEST STATUS CHN1
00C6	STS2	SELF-TEST STATUS CHN2
00C8	STS3	SELF-TEST STATUS CHN3
00CA	CAL0	CALIBRATION STATUS CHN0
00CC	CAL1	CALIBRATION STATUS CHN1
00CE	CAL2	CALIBRATION STATUS CHN2
00D0	CAL3	CALIBRATION STATUS CHN3
00D2	GTS	GENERAL TEST STATUS
00D4	ADCS	ADC TEST STATUS
0100-010A	DACGCOEF0	DAC 0 GAIN COEFFICIENTS [8]
0100		DAC 0 2.50V GAIN COEFFICIENT MSB
0102		DAC 0 2.50V GAIN COEFFICIENT LSB
0104		DAC 0 5.00V GAIN COEFFICIENT MSB
0106		DAC 0 5.00V GAIN COEFFICIENT LSB
0108		DAC 0 10.0V GAIN COEFFICIENT MSB
010A		DAC 0 10.0V GAIN COEFFICIENT LSB
010C		RESERVED
010E		RESERVED
0110-011A	DACGCOEF1	DAC 1 GAIN COEFFICIENTS [8]
0110		DAC 1 2.50V GAIN COEFFICIENT MSB
0112		DAC 1 2.50V GAIN COEFFICIENT LSB
0114		DAC 1 5.00V GAIN COEFFICIENT MSB
0116		DAC 1 5.00V GAIN COEFFICIENT LSB
0118		DAC 1 10.0V GAIN COEFFICIENT MSB
011A		DAC 1 10.0V GAIN COEFFICIENT LSB

Table 3-1 VME Memory Map (Continued)

OFFSET ADDRESS (HEX)	NAME	DESCRIPTION
011C		RESERVED
011E		RESERVED
0120-012A	DACGCOEF2	DAC 2 GAIN COEFFICIENTS [8]
0120		DAC 2 2.50V GAIN COEFFICIENT MSB
0122		DAC 2 2.50V GAIN COEFFICIENT LSB
0124		DAC 2 5.00V GAIN COEFFICIENT MSB
0126		DAC 2 5.00V GAIN COEFFICIENT LSB
0128		DAC 2 10.0V GAIN COEFFICIENT MSB
012A		DAC 2 10.0V GAIN COEFFICIENT LSB
012C		RESERVED
012E		RESERVED
0130-013A	DACGCOEF3	DAC 3 GAIN COEFFICIENTS [8]
0130		DAC 3 2.50V GAIN COEFFICIENT MSB
0132		DAC 3 2.50V GAIN COEFFICIENT LSB
0134		DAC 3 5.00V GAIN COEFFICIENT MSB
0136		DAC 3 5.00V GAIN COEFFICIENT LSB
0138		DAC 3 10.0V GAIN COEFFICIENT MSB
013A		DAC 3 10.0V GAIN COEFFICIENT LSB
013C		RESERVED
013E		RESERVED
0180-018A	DACOCOEF0	DAC 0 OFFSET COEFFICIENTS [8]
0180		DAC 0 2.50V OFFSET COEFFICIENT MSB
0182		DAC 0 2.50V OFFSET COEFFICIENT LSB
0184		DAC 0 5.00V OFFSET COEFFICIENT MSB
0186		DAC 0 5.00V OFFSET COEFFICIENT LSB
0188		DAC 0 10.0V OFFSET COEFFICIENT MSB
018A		DAC 0 10.0V OFFSET COEFFICIENT LSB
018C		RESERVED
018E		RESERVED
0190-019A	DACOCOEF1	DAC 1 OFFSET COEFFICIENTS [8]
0190		DAC 1 2.50V OFFSET COEFFICIENT MSB
0192		DAC 1 2.50V OFFSET COEFFICIENT LSB
0194		DAC 1 5.00V OFFSET COEFFICIENT MSB
0196		DAC 1 5.00V OFFSET COEFFICIENT LSB
0198		DAC 1 10.0V OFFSET COEFFICIENT MSB
019A		DAC 1 10.0V OFFSET COEFFICIENT LSB

Table 3-1 VME Memory Map (Continued)

OFFSET ADDRESS (HEX)	NAME	DESCRIPTION
019C		RESERVED
019E		RESERVED
01A0-01AA	DACOCOEF2	DAC 2 OFFSET COEFFICIENTS [8]
01A0		DAC 2 2.50V OFFSET COEFFICIENT MSB
01A2		DAC 2 2.50V OFFSET COEFFICIENT LSB
01A4		DAC 2 5.00V OFFSET COEFFICIENT MSB
01A6		DAC 2 5.00V OFFSET COEFFICIENT LSB
01A8		DAC 2 10.0V OFFSET COEFFICIENT MSB
01AA		DAC 2 10.0V OFFSET COEFFICIENT LSB
01AC		RESERVED
01AE		RESERVED
01B0-01BA	DACOCOEF3	DAC 3 OFFSET COEFFICIENTS [8]
01B0		DAC 3 2.50V OFFSET COEFFICIENT MSB
01B2		DAC 3 2.50V OFFSET COEFFICIENT LSB
01B4		DAC 3 5.00V OFFSET COEFFICIENT MSB
01B6		DAC 3 5.00V OFFSET COEFFICIENT LSB
01B8		DAC 3 10.0V OFFSET COEFFICIENT MSB
01BA		DAC 3 10.0V OFFSET COEFFICIENT LSB
01BC		RESERVED
01BE		RESERVED
0200-021E	ADCGCOEF0	ADC GAIN/OFFSET COEFFICIENTS [32]
0200		ADC 2.50V GAIN COEFFICIENT MSB
0202		ADC 2.50V GAIN COEFFICIENT LSB
0204		ADC 2.50V OFFSET COEFFICIENT MSB
0206		ADC 2.50V OFFSET COEFFICIENT LSB
0208		ADC 5.00V GAIN COEFFICIENT MSB
020A		ADC 5.00V GAIN COEFFICIENT LSB
020C		ADC 5.00V OFFSET COEFFICIENT MSB
020E		ADC 5.00V OFFSET COEFFICIENT LSB
0210		ADC 10.0V GAIN COEFFICIENT MSB
0212		ADC 10.0V GAIN COEFFICIENT LSB
0214		ADC 10.0V OFFSET COEFFICIENT MSB
0216		ADC 10.0V OFFSET COEFFICIENT LSB
0800-087E	DACRAW0	DAC CALIBRATION DATA FOR DAC 0 [32]
0880-08FE	DACRAW1	DAC CALIBRATION DATA FOR DAC 1 [32]
0900-097E	DACRAW2	DAC CALIBRATION DATA FOR DAC 2 [32]

Table 3-1 VME Memory Map (Continued)

OFFSET ADDRESS (HEX)	NAME	DESCRIPTION
0980-09FE	DACRAW3	DAC CALIBRATION DATA FOR DAC 3 [32]
0A00-0A7E	DACREAL0	DAC CALIBRATION DATA FOR DAC 0 [32]
0A80-0AFE	DACREAL1	DAC CALIBRATION DATA FOR DAC 1 [32]
0B00-0B7E	DACREAL2	DAC CALIBRATION DATA FOR DAC 2 [32]
0B80-0BFE	DACREAL3	DAC CALIBRATION DATA FOR DAC 3 [32]
0C00-0C7E	ADCRAW0	ADC CALIBRATION DATA FOR DAC 0 [32]
0C80-0CFE	ADCRAW1	ADC CALIBRATION DATA FOR DAC 1 [32]
0D00-0D7E	ADCRAW2	ADC CALIBRATION DATA FOR DAC 2 [32]
0D80-0DFE	ADCRAW3	ADC CALIBRATION DATA FOR DAC 3 [32]
0E00-0E7E	ADCREAL0	ADC CALIBRATION DATA FOR DAC 0 [64]
0E80-0EFE	ADCREAL1	ADC CALIBRATION DATA FOR DAC 1 [64]
0F00-0F7E	ADCREAL2	ADC CALIBRATION DATA FOR DAC 2 [64]
0F80-0FFE	ADCREAL3	ADC CALIBRATION DATA FOR DAC 3 [64]

3.2.1 VME Data Memory Map Overview

Table 3-2 VME Data Memory Map Overview

OFFSET ADDRESS (HEX)	DESCRIPTION	SIZE
00000-03FFE	VME CSR DATA AREA	8 Kwords
04000-07FFE	WAVEFORM 8 KWORDS BUFFER #1	8 Kwords
08000-0BFFE	WAVEFORM 8 KWORDS BUFFER #2	8 Kwords
0C000-0FFFE	WAVEFORM 8 KWORDS BUFFER #3	8 Kwords
10000-13FFE	WAVEFORM 8 KWORDS BUFFER #4	8 Kwords
14000-17FFE	WAVEFORM 8 KWORDS BUFFER #5	8 Kwords
18000-1BFFE	WAVEFORM 8 KWORDS BUFFER #6	8 Kwords
1C000-1FFFE	WAVEFORM 8 KWORDS BUFFER #7	8 Kwords
n/a	E^2PROM EPLD DATA	8 Kwords
n/a	E^2PROM EPLD DATA	8 Kwords
n/a	E^2PROM CALIBRATION DATA	8 Kwords
n/a	E^2PROM CALIBRATION DATA	8 Kwords

WAVEFORM MEMORY 7 x 8 Kwords = 56 Kwords
E^2PROM CALIBRATION DATA 2 x 8 Kwords = 16 Kwords
E^2PROM EPLD DOWNLOAD DATA 2 x 8 Kwords = 16 Kwords

NOTE

The DSP accesses the VME RAM in 8 k pages. However, the RAM is fully contiguous from 0x00000 to 0x1FFFE to the VME.

3.3 Register Definitions

3.3.1 Board ID Register (BID)

The Board ID register is a read-only status register, the contents of which identifies the VME-4145. The board ID for the VME-4145 is 0x3C00.

Table 3-3 Board ID Register Bit Map

BOARD ID REGISTER (OFFSET 0x0000) READ-ONLY, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
0	0	1	1	1	1	0	0
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
0	0	0	0	0	0	0	0

3.3.2 DAC Channel Pointer (CHN)

The CHN is a read/write register and is used in conjunction with the firmware commands that control certain functions that are channel dependent. When a command that affects a particular channel is issued, the CHN register points to the corresponding DAC.

NOTE

The user must configure this register before issuing the command.

Table 3-4 DAC Channel Pointer Register Bit Map

DAC CHANNEL POINTER REGISTER (OFFSET 0x0002) READ/WRITE							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
X	X	X	X	X	X	X	X
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
X	X	X	X	X	X	CH1	CH0

x=Don't Care

Bit [1:0] **CH[1:0]** Two-bit binary number from zero to three.

Table 3-5 Two-Bit Binary

CH1	CH0	CHANNEL
0	0	ZERO
0	1	ONE
1	0	TWO
1	1	THREE

NOTE

If CHN is set to two or three on a two channel model and a command is issued, the DSP will reject the command request and set the CRR register to an invalid command code.

3.3.3 VME Command Register (VCR)

The VCR is a read/write register which controls the configuration and operation of the VME-4145.

Table 3-6 VME Command Register Bit Map

VME COMMAND REGISTER (OFFSET 0x0004) READ/WRITE, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
0	0	0	0	0	0	0	0
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0

Bit [7:0] **CC[7:0]** - Command Code. All software control and configuration of the VME-4145 is achieved through this field. The supported command codes are as follows: (Outlined in *"VME-4145 Commands"* and Table 3-23.)

The VCR is a command mailbox between the VME user and the DSP board processor. The DSP will monitor this location until a nonzero value is seen. Once a command is recognized, the DSP will process the command. During this time period, the DSP will take over the VME RAM. Any access to any valid VME-4145 address will result in reading the unit's board ID (0x3C00).

The VME can loop periodically reading the VCR. When it reads back 0x3C00, it must continue to loop. If it reads back the actual command code, this indicates the DSP has not yet started to process the command. If it reads back zero, this indicates the DSP has processed the request. At this point, the user may wish to check the response register for acknowledgment of the command.

Example:
```
while (c = uut-> vcr!= Fnull) {spin}
where: Fnull = 0
c = unsigned short
uut = structure defining VME-4145 memory map
vcr = VME Command Register element of uut
```

Many of the VCR commands use multiple associated registers during the execution of a command. These registers must be set before the VCR is loaded with a command. The most used registers are the Command Data Register (CDR), and the CMD_DATA_LSW and CMD_DATA_MSW auxiliary data registers.

These general-purpose registers are used to pass command parameters to or from the DSP. See each command below for details on the use of these registers. The CHN register is the DAC channel pointer. When issuing a channel-specific command, the CHN register must be set to the desired channel.

After the command has been processed by the DSP, the DSP will clear the VCR to zero (NULL command). The DSP will also set the CRR command response register with a response code. See Table 3-9 for a listing of response codes. The most used responses are (1) VALID COMMAND and (2) INVALID COMMAND.

3.3.4 Command Data Registers (CDRs)

Table 3-7 Command Data Register Descriptions

VME	NAME	DESCRIPTION
0006	CDR	CMD DATA REG
0016	CMD_DATA_LSW	AUXILIARY DATA REG LSW
0018	CMD_DATA_MSW	AUXILIARY DATA REG MSW

NOTE

These are general-purpose registers used to pass a 16-bit value with a command in DSP firmware. See individual commands for any usage of this register.

3.3.5 Command Response Register (CRR)

The CRR is a read/write register which indicates success, failure, or status of the most recent Command.

Table 3-8 Command Response Register Bit Map

COMMAND RESPONSE REGISTER (OFFSET 0x0008) READ/WRITE, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0

Bit [15:0] **CR[15:0]** - This code indicates the response of the Local DSP to the most recent command and is interpreted as shown in Table 3-9 below.

Table 3-9 Command Response Register Code Identification

DESCRIPTION	CODE
Valid Command Recognized and Processed	0x0001
Invalid, Command not Recognized	0x0002
Power Up Boot is Completed	0x0010
Waveform Loaded	0x3
Calibrated Waveform Loaded	0x4
Calibration Waveform Loaded, Clipping Occurred	0x5

3.3.6 General Configuration Register (GCR)

The GCR is read/write. The user may read this location to determine the past history of its state. Writing a new value will not take effect until after the VCR command is issued.

NOTE

The GCR must be set before issuing a load waveform command. The DSP must know what data format you are using before you issue a 0x6 or 0x23 load waveform command.

Table 3-10 General Configuration Register Bit Map

GENERAL CONFIGURATION REGISTER (OFFSET 0x000A) READ/WRITE, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
ADC SKIP CAL	ADC SKIP VER	0	0	0	0	0	0
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
0	0	0	0	BLINK	LED OFF	DATA FMT	CAL MODE

Bit 15 **ADC SKIP CAL** - (FRR = Version 01.0a and above) A logical "1" causes the ADC calibration to be skipped during a calibration command. A logical "0" causes the ADC calibration to be included during a calibration command.

Bit 14 **ADC SKIP VER** - (FRR = Version 01.0a and above) A logical "1" causes the ADC verification to be skipped during a calibration command. A logical "0" causes the ADC verification to be included during a calibration command.

NOTE

The ADC has to have been calibrated and gain and offset coefficients created in order for these two commands to work properly. For example, if the ADC has been previously calibrated, it can then be verified and the DACs calibrated without recalibration of the ADC.

NOTE

If calibration coefficients have been stored, it is possible to recall them and verify their accuracy without recalibrating the board.

Bit 3 **BLINK** - (FRR = Version 1.7 and above) A logical "0" causes the DSP to blink the front panel LED during many of its operations. This visual feedback is useful during software development. A logical "1" disables the blinking during speed critical operations (such as downloading a waveform). Applications requiring fast execution should set this bit. On power up this bit is set.

Bit 2 **LED OFF** - A logical "1" causes the front panel status LED to be turned Off. A logical "0" causes the front panel status LED to be turned On. This bit is set to a logic "0" under firmware control in the event of a self-test/calibration failure. This bit is set to a logic "1" under firmware control in the event of a self-test/calibration without failures.

Bit 1 **DATA FMT** - A logical "1" causes the data in the Waveform Buffer to be interpreted in offset binary format for bipolar ranges or binary format for unipolar ranges. A logic "0" causes the data to be interpreted as two's complement format (default is logic "0"). This causes the loaded data to be stored in the proper format.

Bit 0 **CAL Mode** - A logical "1" causes the DSP to overwrite the uncalibrated waveform buffer data with the new calibrated values during a VCR = 0x23 load full waveform table with Gain/Offset Autocalibration. This allows the user to see the actual corrected values given to the DAC. A logical "0" will cause the DSP to leave uncorrected user data in the local VME waveform buffer. Corrected data will still be written to the DAC engine RAM, but will not be echoed locally. This allows users to load a waveform once, and then download the waveform to multiple channels without reloading the original.

3.3.7 Self-Test Status Register (SSR)

The SSR is a read-only status register which indicates any channels which have failed to pass power up, reset, or software-initiated self-test/calibration. If any of the channels have failed, the user may examine one of the channel dependent status registers for further details. Calibration will set to zero any bits previously set during power up testing before calibration begins.

Table 3-11 Self-Test Status Register Bit Map

SELF-TEST STATUS REGISTER (OFFSET 0x000C) READ-ONLY, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
0	0	0	0	0	0	0	0
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
0	0	0	0	FAIL CH3	FAIL CH2	FAIL CH1	FAIL CH0

Bit [3:0] **Fail CH[3:0]** - A logical "1" indicates that the corresponding channel has failed self-test or has failed to calibrate properly. A logical "0" indicates that the corresponding channel has passed self-test or calibration. For the VME-4145-001, Fail CH[3:2] will always be logical "0."

3.3.8 Software Reset (RST)

This register is set to 0x1D whenever the board is reset by either software reset or power up. The user may use this flag to detect a reset condition. This register is cleared to zero if one of the following operations are performed:

1. The board is instructed to perform auto-calibration
2. Downloading of calibration coefficients from EEPROM to VME RAM.

The user can use this register as a flag to indicate if the coefficients exist, so that calibrated waveforms can be loaded.

3.3.9 Firmware Revision Register (FRR)

The FRR is a read-only status register indicating the firmware revision level. Both major and minor revision levels are indicated and are interpreted as version.

Table 3-12 Firmware Revision Register Bit Map

FIRMWARE REVISION REGISTER (OFFSET 0x001E) READ-ONLY, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
MAR7	MAR6	MAR5	MAR4	MAR3	MAR2	MAR1	MAR0
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
MIR7	MIR6	MIR5	MIR4	MIR3	MIR2	MIR1	MIR0

Bits [15:8] **MAR[7:0]** - Major Revision. Hexadecimal code indicating the "major" revision level from 0 to 255.

Bits [7:0] **MIR[7:0]** - Minor Revision. Hexadecimal code indicating the "minor" revision level from 0 to 255.

Example: $0102 is interpreted as version 1.02.

3.3.10 Channel Status Register (CSR0 - CSR3)

These four registers indicate the status of the EPLD 'engine' waveform generators. A 0x0011 report EPLD status command must be issued to update these registers. See *"Command: Report EPLD Engine Status Command"* .

Table 3-13 Channel Status Register Bit Map

CHANNEL STATUS REGISTER BIT MAP (Offset 0x0020-0x0026) READ/WRITE, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
0	0	0	0	0	0	0	0
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
0	0	0	0	SREQ	CSA2	CSA1	CSA0

3.3.11 Channel Status Register Bit Definition

Bit 3 **SREQ** - Software request tells the user that the trigger has occurred and a waveform is running. The SREQ can then be cleared by writing to the Ending Address Register (EAR). This flag is used for chaining segmented waveforms together. The DSP must monitor this bit and clear it after it has updated the SAR and EAR registers in real time.

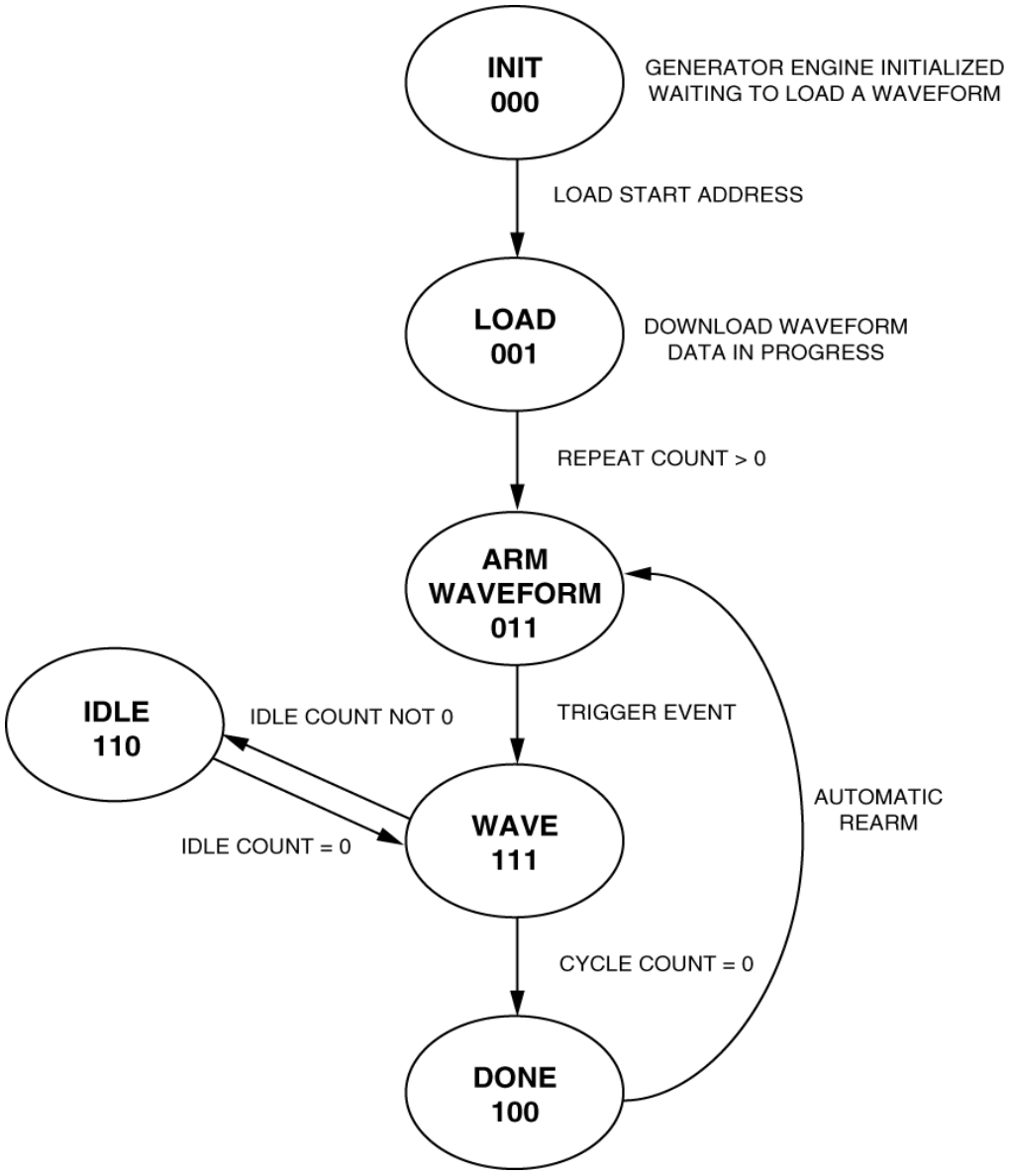
Bits [2:0] **CSA[2:0]** -Engine state machine status. The status codes are detailed in Table 3-14.

Table 3-14 Engine State Machine Code Identification

CSA	NAME	STATE MACHINE STATUS
000	INIT	INITIALIZED STATE
001	LOAD	LOADING WAVEFORM STATE
010	N/A	
011	ARM	TRIGGER IS ARMED AND READY
100	DONE	WAVEFORM FINISHED OR HALTED
101	N/A	
110	IDLE	IDLE WAIT BETWEEN WAVEFORMS
111	WAVE	WAVEFORM RUNNING

Refer to the EPLD state machine diagram below for further information on the flow of the state machine.

Figure 3-1 EPLD State Machine Diagram



NOTE

This is a very simplified diagram. Many conditional paths exist between states at higher level of detail.

3.3.12 Channel Configuration Registers (CCR0 - CCR3)

Table 3-15 Channel Configuration Registers Bit Map

CHANNEL CONFIGURATION REGISTERS (OFFSET 0x0028-0x002E) READ/WRITE, BYTE/WORD							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
DAC SKIP CAL	DAC SKIP VER	0	Bipolar	0	0	0	Online
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
Reserved	Mode2	Mode1	Mode0	SYNC	Trig2	Trig1	Trig0

BIT 15 **DAC SKIP CAL -** (FRR = Version 01.0a and above) A logical "1" causes the DAC calibration to be skipped during a calibration command. A logical "0" causes the DAC calibration to be included during a calibration command.

BIT 14 **DAC SKIP VER** - (FRR = Version 01.0a and above) A logical "1" causes the DAC verification to be skipped during a calibration command. A logical "0" causes the DAC verification to be included during a calibration command.

NOTE

The DAC has to have been calibrated and gain and offset coefficients created in order for these two commands to work properly. For example, if the DAC has been previously calibrated, it can then be verified without recalibrating. If calibration coefficients have been stored, it is possible to recall them and verify their accuracy without recalibrating the board.

Bit 12 **Bipolar** - A logical "0" indicates a unipolar output voltage. A logical"1" selects a bipolar output voltages.

Bit 8 **Online** - A logical "1" disables the analog output. A logical "0" enables the analog output.

Bit 7 **Reserved** - Write to zero.

Bits [6:4] **Mode [2:0]** - These bits are used in controlling the analog output range in accordance with Table 3-16.

Table 3-16 Analog Output Range Settings

BIPOLAR	MODE 2	1	0	GAIN	BIPOLAR JUMPER	DAC RANGE
1	1	1	0	x1	IN	-2.5 to +2.5 VDC
0	1	1	0	x2	OUT	0 to +5 VDC
1	1	0	1	x2	IN	-5 to +5 VDC
0	1	0	1	x4	OUT	0 to +10 VDC
1	0	1	1	x4	IN	-10 to +10 VDC

Incorrectly setting these three bits may cause an output to rail to ±12 VDC.

Bit 3 **SYNC** - A logical "0" indicates active low output sync pulse (one sample period). A logical "1" indicates active high output sync pulse (one sample period).

Bits [2:0] **TRIG [2:0]** - These bits control the type of trigger to be selected. The description of each bit is shown in Table 3-17.

Table 3-17 Trigger Selections

TRIG	DESCRIPTION
001	RISING EDGE (SYNCHRONIZED) OR SOFTWARE
000	FALLING EDGE (SYNCHRONIZED) OR SOFTWARE
010	LOGICAL ZERO (WITH PROPER SET UP/HOLD)
011	LOGICAL ONE (WITH PROPER SET UP/HOLD)
100	EXTERNAL TRIGGERS OFF, SOFTWARE TRIGGER ONLY
101	EXTERNAL TRIGGERS OFF, SOFTWARE TRIGGER ONLY
110	EXTERNAL TRIGGERS OFF, SOFTWARE TRIGGER ONLY
111	EXTERNAL TRIGGERS OFF, SOFTWARE TRIGGER ONLY

3.3.13 Internal Time Base Registers

Table 3-18 Internal Time Base Register Bit Map

SRH0-3 (0X30-3C)															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
X	X	X	X	X	X	X	X	X	X	X	X	---------MSB----------			
SRL0-3 (0X32-3E)															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
----------LSB----------															

The user loads the generator with a 20-bit value N, where:

F => N => 2^20-1

A lower limit of 0x10 defines the maximum clock rate at which the unit is designed to run (2.5 Ms/s). Each bit represents the internal time base clock which provides the 25 ns per tick resolution.

Example: 16 * 25 ns = 400 ns

NOTE

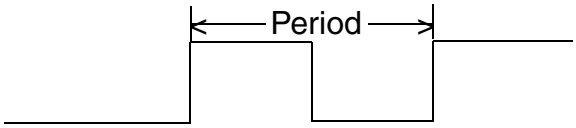
This register may be reloaded while a waveform is running. In this way, a software-controlled frequency sweep may be implemented. There will be a slight delay between writing the 0x9 command and the actual frequency change. This is due to the DSP processing time to interpret the command.

There are two factors which will determine the frequency of the waveform that is generated. The first is the number of samples used to determine a single cycle or period of the waveform. The second factor is how fast each data point in the waveform is clocked out.

Example 1: Generating a 1kHz square wave.

Frequency (F), Hz = 1 / Period (P), sec

Where period (P) is equal to the time it takes to generate one complete cycle of the waveform.



P = Ns x Sr

Where: P =Period
Ns=Number of Samples Loaded for One Period
Sr =Sample Rate

The sample rate, Sr is set by programming the SRH0/SRL0 20-bit value.

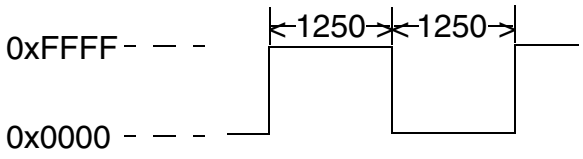
Let N = 20-bit value loaded into the SRH0/SRL0 registers.
The internal clock is 40 MHz, or 25 ns per tick

Therefore:

Sr = N x 25 ns

Let N = 0x0000f, or decimal 15

Ns= 2500, 1250 samples @ 0xFFFF followed by 1250 samples @ 0x0000



Therefore: Sr = N x 25ns = 16 x 25E-9 = 400 ns

P = Ns x Sr = 2500 x 400 ns = 1e-3

it takes one millisecond to complete a single cycle.

And finally,

F = 1 / P = 1 / 1e-3 or 1 kHz

To combine it all together:

$$F = \frac{1}{\text{\# of Samples per Cycle} \times 25\ e^{-}9 \times (\text{[SRH0 : SRL0]} + 1)}$$

Example 2: Generating a 10.0 kHz Sine wave with a maximum DAC output rate of 2.5 million samples per second.

1 / 2.5e+6 = 400 ns/sample

The VME-4145 runs at 25 ns/tick on the internal clock.

400 ns / 25 ns/tick = 16 clock ticks required per sample

Program the sample rate registers as follows:

SRH0 = 0x0000

SRL0 = 0x000f

F = 2.5e+6 / Ns

Where:Ns = number of samples in one cycle

10000 Hz = (2.5e+6 / Ns)

Solving for Ns gives,

Ns = 2.5e+6 / 10e+3 = 250 Samples

Therefore, a sine waveform, 0 to 2π (6.28 radians), must be downloaded in 250 samples.

6.28 Radians / 250 Samples = 25.1e-3 Radians per Sample Point

or

360° / 250 Samples = 1.44° per Sample Point

When downloading values, convert the 0 to ±1 SIN values to 0x0000 to 0xFFFF HEX values, where 0 = 0x8000 for bipolar outputs.

3.3.14 Idle Time Registers

Table 3-19 Idle Time Register Bit Map

IH0-3 (0X0040-0X004C)															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
X	X	X	X	X	X	X	X	----------MSB----------							
IL0-3 (0X0042-0X004E)															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
----------LSB----------															

3.3.15 Starting Address Pointer Registers

Table 3-20 Starting Address Pointer Register Bit Map

DSRH0-3															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	MSB

DSRL0-3															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
----------LSB----------															

3.3.16 Ending Address Pointer Registers

Table 3-21 Ending Address Pointer Register Bit Map

DERH0-3															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	MSB

DERL0-3															
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit09	Bit08	Bit07	Bit06	Bit05	Bit04	Bit03	Bit02	Bit01	Bit00
----------LSB----------															

3.3.17 Clock MUX Register

Table 3-22 Clock MUX Register Bit Map

CLOCK MUX REGISTER, READ/WRITE, BYTE/WORD (0X00C0)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
X	COM D2	COM D1	COM D0	CHN3 D2	CHN3 D1	CHN3 D0	DHN2 D2
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
CHN2 D1	CHN2 D0	CHN1 D2	CHN1 D1	CHN1 D0	CHN0 D2	CHN0 D1	CHN0 D0

These three-bit nibbles define the clock source for each channel's waveform output. The D2-D0 bits for each channel are defined in Table 3-23 on page 66.

Table 3-23 Clock MUX Code Identification

CHNX D2-D0	DESCRIPTION
000	DEDICATED INTERNAL TIMEBASE FOR CHANNELS (0, 1, 2, 3)
001	FRONT PANEL
010	INVERTED LOGIC OF FRONT PANEL
011	COMMON SOURCE (SEE BELOW)
100	FORCE CLOCK LOW
101	FORCE CLOCK HIGH
110	N/A (WILL FORCE CLOCK HIGH)
111	N/A (WILL FORCE CLOCK HIGH)

The unit is designed to allow the use of a common clock to provide synchronized outputs for multiple channels. The source of the common clock is also programmable. The programming sequence is described in Table 3-24 on page 66.

Table 3-24 Common Clock Code Identification

COM D2-D0	DESCRIPTION
000	INTERNAL TIMEBASE FOR CHANNEL 0
001	FRONT PANEL CHN0
010	INVERTED LOGIC OF FRONT PANEL
011	INTERNAL TIMEBASE CHANNEL 0 DIVIDED BY TEN
100	FORCE CLOCK LOW
101	FORCE CLOCK HIGH
110	N/A (WILL FORCE CLOCK HIGH)
111	N/A (WILL FORCE CLOCK HIGH)

3.3.18 Self-Test Status Registers (STS0 - STS3)

Offset = 0x00C2 - 0x00C8

These registers are set after a power-on reset or self-test command is executed. A nonzero value loaded into one of these registers indicates the test software found an error. These four registers indicate channel-specific errors. An error indicated in one channel should not affect another channel's operation. STS0, STS1, STS2, and STS3 indicates channel 0, channel 1, channel 2, and channel 3 error information, respectively. The following table is a list of the status values returned.

Table 3-25 Hardware Failure Codes for Self-Test Status

Status Code	Hardware Failure	Status Code	Hardware Failure
F	CH0 waveform RAM data = address test failed	2D	CH2 waveform RAM data = address test failed
10	Engine failed to initialize during CH0 waveform memory RAM AED test	2E	Engine failed to initialize during CH2 waveform memory RAM AED test
11	Engine failed to reach load state during CH0 waveform memory RAM AED	2F	Engine failed to reach load state during CH2 waveform memory RAM AED
12	CH0 waveform RAM data = FFFF test failed	30	CH2 waveform RAM data = FFFF test failed
13	Engine failed to initialize during CH0 waveform memory RAM FFFF test	31	Engine failed to initialize during CH2 waveform memory RAM FFFF test
14	Engine failed to reach load state during CH0 waveform memory RAM FFFF	32	Engine failed to reach load state during CH2 waveform memory RAM FFFF
15	CH0 waveform RAM data = AAAA test failed	33	CH2 waveform RAM data = AAAA test failed
16	Engine failed to initialize during CH0 waveform memory RAM AAAA test	34	Engine failed to initialize during CH2 waveform memory RAM AAAA test
17	Engine failed to reach load state during CH0 waveform memory RAM AAAA	35	Engine failed to reach load state during CH2 waveform memory RAM AAAA
18	CH0 waveform RAM data = 5555 test failed	36	CH2 waveform RAM data = 5555 test failed
19	Engine failed to initialize during CH0 waveform memory RAM 5555 test	37	Engine failed to initialize during CH2 waveform memory RAM 5555 test
1A	Engine failed to reach load state during CH0 waveform memory RAM 5555	38	Engine failed to reach load state during CH2 waveform memory RAM 5555
1B	CH0 waveform RAM data = 0000 test failed	39	CH2 waveform RAM data = 0000 test failed
1C	Engine failed to initialize during CH0 waveform memory RAM 0000 test	3A	Engine failed to initialize during CH2 waveform memory RAM 0000 test
1D	Engine failed to reach load state during CH0 waveform memory RAM 0000	3B	Engine failed to reach load state during CH2 waveform memory RAM 0000
1E	CH1 waveform RAM data = address test failed	3C	CH3 waveform RAM data = address test failed
1F	Engine failed to initialize during CH1 waveform memory RAM AED test	3D	Engine failed to initialize during CH3 waveform memory RAM AED test
20	Engine failed to reach load state during CH1 waveform memory RAM AED	3E	Engine failed to reach load state during CH3 waveform memory RAM AED
21	CH1 waveform RAM data = FFFF test failed	3F	CH3 waveform RAM data = FFFF test failed
22	Engine failed to initialize during CH1 waveform memory RAM FFFF test	40	Engine failed to initialize during CH3 waveform memory RAM FFFF test
23	Engine failed to reach load state during CH1 waveform memory RAM FFFF	41	Engine failed to reach load state during CH3 waveform memory RAM FFFF
24	CH1 waveform RAM data = AAAA test failed	42	CH3 waveform RAM data = AAAA test failed
25	Engine failed to initialize during CH1 waveform memory RAM AAAA test	43	Engine failed to initialize during CH3 waveform memory RAM AAAA test
26	Engine failed to reach load state during CH1 waveform memory RAM AAAA	44	Engine failed to reach load state during CH3 waveform memory RAM AAAA
27	CH1 waveform RAM data = 5555 test failed	45	CH3 waveform RAM data = 5555 test failed
28	Engine failed to initialize during CH1 waveform memory RAM 5555 test	46	Engine failed to initialize during CH3 waveform memory RAM 5555 test

Table 3-25 Hardware Failure Codes for Self-Test Status (Continued)

Status Code	Hardware Failure	Status Code	Hardware Failure
29	Engine failed to reach load state during CH1 waveform memory RAM 5555	47	Engine failed to reach load state during CH3 waveform memory RAM 5555
2A	CH1 waveform RAM data = 0000 test failed	48	CH3 waveform RAM data = 0000 test failed
2B	Engine failed to initialize during CH1 waveform memory RAM 0000 test	49	Engine failed to initialize during CH3 waveform memory RAM 0000 test
2C	Engine failed to reach load state during CH1 waveform memory RAM 0000	4A	Engine failed to reach load state during CH3 waveform memory RAM 0000

AED =Address Equals Data

3.3.19 DAC Calibration Status Registers (CAL0-CAL3)

Offset = 0x00CA - 0x00D0

These registers are set after a calibration command has been executed. A nonzero value loaded into one of these registers indicates that the DAC calibration was not successful for that channel. The following table is a list of the status values returned. These registers will not be updated if the skip DAC verification bit has been set for that channel.

Table 3-26 DAC Hardware Failure Codes for Calibration Status

Status Code	DAC Hardware Failure
0x000D	DAC Engine Verification Error during Calibration
0x0100	DAC Verification Failure during waveform loading of channel 0
0x0101	DAC Verification Failure during waveform loading of channel 1
0x0102	DAC Verification Failure during waveform loading of channel 2
0x0103	DAC Verification Failure during waveform loading of channel 3
0x2000	DAC RAM Verification Error during Calibration
0x1000	DAC Engine Verification Error during Calibration
0x4002	DAC Verification Failure 5.00V range
0x4004	DAC Verification Failure 10.0V range
0x2000	DAC RAM Verification Error during Calibration
0x1000	DAC Engine Verification Error during Calibration

3.3.20 General Test Status Register (GTS)

Offset = 0x00D2This register indicates a board failure of a general nature. An error will affect the operation of all channels. This register gets set after self-test. A non-zero value indicates an error condition has been detected.

Table 3-27 Hardware Failure Codes for General Test Status

Status Code	Hardware Failure
0	General failure not detected
1	Engine failed to initialize during waveform load
2	Engine failed to reach load state during waveform load
3	Engine failed to initialize during configuration
4	Engine failed to reach load state during configuration
5	Engine failed to initialize during waveform dump
6	Engine failed to reach load state during waveform dump
7	VMERAM data = address equals data test failed
8	VMERAM data = FFFF test failed
9	VMERAM data = AAAA test failed
A	VMERAM data = 5555 test failed
B	VMERAM data = 0000 test failed
C	Write bit routine failed during flex loading
D	Engine failed to initialize during power up configuration
E	Engine failed to reach load state during power up configuration

3.3.21 ADC Calibration Status Register (ADCS)

Offset = 0x00D4

This register is set after a calibration command has been executed. A nonzero value loaded into this register indicates that the ADC calibration was not successful. The following table is a list of the status values returned. This register will not be updated if the skip ADC verification bit has been set.

Table 3-28 ADC Hardware Failure Codes for Calibration Status

Status Code	ADC Hardware Failure
0x8001	ADC Verification Failure 2.50 V range
0x8002	ADC Verification Failure 5.00 V range
0x8004	ADC Verification Failure 10.0 V range

3.3.22 DAC Gain Coefficient Registers for Channel 0 (GCOEF0)

Offset = 0x0100 - 0x010A

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC gain coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the gain correction necessary to achieve a calibrated output for that range. The ideal value (0x7d700000) represents a gain of approximately 0.98. The maximum value (0x7ffffff) represents a gain of 1.0. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.23 DAC Offset Coefficient Registers for Channel 0 (OCOEF0)

Offset = 0x0180 - 0x018A

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC offset coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the offset correction necessary to achieve a calibrated output for that range. The ideal value (0x00000000) represents a dc offset of zero counts. This value may be either positive or negative. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.24 DAC Gain Coefficient Registers for Channel 1 (GCOEF1)

Offset = 0x0110 - 0x011A

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC gain coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the gain correction necessary to achieve a calibrated output for that range. The ideal value (0x7d700000) represents a gain of approximately 0.98. The maximum value (0x7ffffff) represents a gain of 1.0. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.25 DAC Offset Coefficient Registers for Channel 1 (OCOEF1)

Offset = 0x0190 - 0x019A

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC offset coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the gain correction necessary to achieve a calibrated output for that range. The ideal value (0x00000000) represents a dc offset of zero counts. This value may be either positive or negative. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.26 DAC Gain Coefficient Registers for Channel 2 (GCOEF2)

Offset = 0x0120 - 0x012A

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC gain coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the gain correction necessary to achieve a calibrated output for that range. The ideal value (0x7d700000) represents a gain of approximately 0.98. This maximum value

(0x7ffffff) represents a gain of 1.0. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.27 DAC Offset Coefficient Registers for Channel 2 (OCOEF2)

Offset = 0x01A0 - 0x01AA

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC offset coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the offset correction necessary to achieve a calibrated output for that range. The ideal value (0x00000000) represents a dc offset of zero counts. This value may be either positive or negative. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.28 DAC Gain Coefficient Registers for Channel 3 (GCOEF3)

Offset = 0x0130 - 0x013A

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC gain coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the gain correction necessary to achieve a calibrated output for that range. The ideal value (0x7d700000) represents a gain of approximately 0.98. This maximum value (0x7ffffff) represents a gain of 1.0. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.29 DAC Offset Coefficient Registers for Channel 3 (OCOEF3)

Offset = 0x01B0 - 0x01BA

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC offset coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the offset correction necessary to achieve a calibrated output for that range. The ideal value (0x00000000) represents a dc offset of zero counts. This value may be either positive or negative. These registers will not be updated if the skip DAC calibration bit has been set.

3.3.30 ADC Gain Coefficient Registers (ADCGCOEF)

Offset = 0x0200 - 0x021E

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The ADC gain coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the gain correction necessary to achieve a calibrated input for that range. The ideal value (0x7d700000) represents a gain of approximately 1.10. The maximum value (0x7ffffff) represents a gain of 1.0. These registers will not be updated if the skip ADC calibration bit has been set.

3.3.31 ADC Offset Coefficient Registers (ADCOCOEF)

Offset = 0x0200 - 0x021E

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The ADC offset coefficient registers contain three 32-bit values, one for each of the three ranges. The values represent the offset correction necessary to achieve a calibrated input for that range. The ideal

value (0x00000000) represents a dc offset of zero counts. This value may be either positive or negative. These registers will not be updated if the skip ADC calibration bit has been set.

3.3.32 DAC Gain Coefficient Registers (GCOEF)

Offset = 0x02A0 - 0x02A2

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC gain coefficient registers contain a 32-bit value. The value stored is for the last operation completed. The values represent the gain correction necessary to achieve a calibrated output for that operation. The ideal value (0x7d700000) represents a gain of approximately 0.98. The maximum value (0x7ffffff) represents a gain of 1.0.

3.3.33 DAC Offset Coefficient Registers (OCOEF1)

Offset = 0x02A4 - 0x02A6

These registers are set after a calibration command has been executed or coefficients recalled from EEPROM. The DAC offset coefficient registers contain a 32-bit value. The value stored is for the last operation completed. The values represent the offset correction necessary to achieve a calibrated output for that operation. The ideal value (0x00000000) represents a dc offset of zero counts. The value may be either positive or negative.

3.3.34 DAC Calibration Raw Data Storage Area [64] (DACRAW0)

Offset = 0x0800 - 0x087E

These registers are used by the DSP during the calibration of channel 0. This data stored here is the 32 raw values that are used during the calibration of the DAC and associated gain circuit.

These 32 values range from:

0x07e0 (-9.84V) to 0xf820 (+9.84V)

These 16-bit values represent the typical values that the user would enter as waveform data.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients are not applied to these values.

3.3.35 DAC Calibration Raw Data Storage Area [64] (DACRAW1)

Offset = 0x0880 - 0x08FE

These registers are used by the DSP during the calibration of channel 1. This data stored here is the 32 raw values that are used during the calibration of the DAC and associated gain circuit.

These 32 values range from:

0x07e0 (-9.84V) to 0xf820 (+9.84V)

These 16-bit values represent the typical values that the user would enter as waveform data.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients are not applied to these values.

3.3.36 DAC Calibration Raw Data Storage Area [64] (DACRAW2)

Offset = 0x0900 - 0x097E

These registers are used by the DSP during the calibration of channel 2. This data stored here is the 32 raw values that are used during the calibration of the DAC and associated gain circuit.

These 32 values range from:

0x07e0 (-9.84V) to 0xf820 (+9.84V)

These 16-bit values represent the typical values that the user would enter as waveform data.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients are not applied to these values.

3.3.37 DAC Calibration Raw Data Storage Area [64] (DACRAW3)

Offset = 0x0980 - 0x09FE

These registers are used by the DSP during the calibration of channel 3. This data stored here is the 32 raw values that are used during the calibration of the DAC and associated gain circuit.

These 32 values range from:

0x07e0 (-9.84V) to 0xf820 (+9.84V)

These 16-bit values represent the typical values that the user would enter as waveform data.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients are not applied to these values.

3.3.38 DAC Calibration Real Data Storage Area [64] (DACREAL0)

Offset = 0x0A00 - 0x0A7E

These registers are used by the DSP during the verification of channel 0. This data stored here is the 32 real values that are used during the verification of the DAC and associated gain circuit.

These 16-bit values represent the typical values that the user would see echoed into the waveform buffer if the echo bit was set in the General Configuration Register.

These 16-bit values are inverted prior to the application of the gain and offset coefficients because there is an inversion in the DAC output circuitry. This data inversion makes the voltage inversion in the output circuitry invisible to the end user.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients have been applied to create these values.

3.3.39 DAC Calibration Real Data Storage Area [64] (DACREAL1)

Offset = 0x0A80 - 0x0AFE

These registers are used by the DSP during the verification of channel 1. This data stored here is the 32 real values that are used during the verification of the DAC and associated gain circuit.

These 16-bit values represent the typical values that the user would see echoed into the waveform buffer if the echo bit was set in the General Configuration Register.

These 16-bit values are inverted prior to the application of the gain and offset coefficients because there is an inversion in the DAC output circuitry. This data inversion makes the voltage inversion in the output circuitry invisible to the end user.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients have been applied to create these values.

3.3.40 DAC Calibration Real Data Storage Area [64] (DACREAL2)

Offset = 0x0B00 - 0x0B7E

These registers are used by the DSP during the verification of channel 2. This data stored here is the 32 real values that are used during the calibration of the DAC and associated gain circuit.

These 16-bit values represent the typical values that the user would see echoed into the waveform buffer if the echo bit was set in the General Configuration Register.

These 16-bit values are inverted prior to the application of the gain and offset coefficients because there is an inversion in the DAC output circuitry. This data inversion makes the voltage inversion in the output circuitry invisible to the end user.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients have been applied to create these values.

3.3.41 DAC Calibration Real Data Storage Area [64] (DACREAL3)

Offset = 0x0B80 - 0x0BFE

These registers are used by the DSP during the verification of channel 3. This data stored here is the 32 real values that are used during the verification of the DAC and associated gain circuit.

These 16-bit values represent the typical values that the user would see echoed into the waveform buffer if the echo bit was set in the General Configuration Register.

These 16-bit values are inverted prior to the application of the gain and offset coefficients because there is an inversion in the DAC output circuitry. This data inversion makes the voltage inversion in the output circuitry invisible to the end user.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients have been applied to create these values.

3.3.42 ADC Calibration Raw Data Storage Area [64] (ADCRAW0)

Offset = 0x0C00 - 0x0C7E

These registers are used by the DSP during the calibration of channel 0. This data stored here is the average of 128 readings taken of the 32 raw values that were used during the calibration of the DAC and associated gain circuit.

The data from these ADC readings were used to develop the gain and offset coefficients during the calibration procedure.

These 32-bit values are stored in the two's complement format. DAC gain and offset coefficients are not applied to these values.

3.3.43 ADC Calibration Raw Data Storage Area [64] (ADCRAW1)

Offset = 0x0C80 - 0x0CFE

These registers are used by the DSP during the calibration of channel 1. This data stored here is the average of 128 readings taken of the 32 raw values that were used during the calibration of the DAC and associated gain circuit.

The data from these ADC readings were used to develop the gain and offset coefficients during the calibration procedure.

These 32-bit values are stored in the two's complement format. DAC gain and offset coefficients are not applied to these values.

3.3.44 ADC Calibration Raw Data Storage Area [64] (ADCRAW2)

Offset = 0x0D00 - 0x0DFE

These registers are used by the DSP during the calibration of channel 2. This data stored here is the average of 128 readings taken of the 32 raw values that were used during the calibration of the DAC and associated gain circuit.

The data from these ADC readings were used to develop the gain and offset coefficients during the calibration procedure.

These 32-bit values are stored in the two's complement format. DAC gain and offset coefficients are not applied to these values.

3.3.45 ADC Calibration Raw Data Storage Area [64] (ADCRAW3)

Offset = 0x0D80 - 0x0DFE

These registers are used by the DSP during the verification of channel 3. This data stored here is the average of 128 readings taken of the 32 raw values that were used during the calibration of the DAC and associated gain circuit.

The data from these ADC readings were used to develop the gain and offset coefficients during the calibration procedure.

These 32-bit values are stored in the two's complement format. DAC gain and offset coefficients are not applied to these values.

3.3.46 ADC Calibration Real Data Storage Area [64] (ADCREAL0)

Offset = 0x0E00 - 0x0E7E

These registers are used by the DSP during the verification of channel 0. The data stored here is the average of 128 readings taken of the 32 corrected values that were output by the DAC during the verification of the DAC and associated gain circuit.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients were applied to the DAC outputs prior to making these readings.

3.3.47 ADC Calibration Real Data Storage Area [64] (ADCREAL1)

Offset = 0x0E80 - 0x0EFE

These registers are used by the DSP during the verification of channel 1. The data stored here is the average of 128 readings taken of the 32 corrected values that were output by the DAC during the verification of the DAC and associated gain circuit.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients were applied to the DAC outputs prior to making these readings.

3.3.48 ADC Calibration Real Data Storage Area [64] (ADCREAL2)

Offset = 0x0F00 - 0x0F7E

These registers are used by the DSP during the verification of channel 2. The data stored here is the 32 real values that are used during the calibration of the DAC and associated gain circuit.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients have been applied to create these values.

3.3.49 ADC Calibration Real Data Storage Area [64] (ADCREAL3)

Offset = 0x0F80 - 0x0FFE

These registers are used by the DSP during the verification of channel 3. The data stored here is the 32 real values that are used during the verification of the DAC and associated gain circuit.

These 16-bit values are stored in the offset binary format. DAC gain and offset coefficients have been applied to create these values.

3.4 Segmented Waveform Programming

The waveform generator allows the storing of multiple waveforms in the generator memory. Since the DSP must perform autocorrection on any data the user enters, it is not capable of high speed on the fly loading of new waveforms. For this reason, the VME-4145 is equipped with a very large waveform storage area.

The user may program in a number of waveforms before waveform generation starts. The user may switch between these under software control or link these together in any fashion they choose. In this way, a single waveform may be selected and generated or up to 64 waveforms can be linked together. If they are linked together, after the first waveform data table has been sequenced through and generated, the DSP will automatically load the next waveform table for generation. This is a seamless process from the standpoint of the output connector.

After the last value is generated from the first table, the first value of the second table will be generated at the next output clock cycle.

Segmentation memory has been set aside to allow programming the segment starting, ending, and next segment to generate link pointers. Each of the four output channels has a dedicated memory area as can be seen in Table 3-29.

Table 3-29 Segmentation Pointer Descriptions

VME	DESCRIPTION	SIZE
0x1000-0x13FE	CHN 0 SEGMENTATION POINTERS	512 WORDS
0x1400-0x17FE	CHN 1 SEGMENTATION POINTERS	512 WORDS
0x1800-0x1BFE	CHN 2 SEGMENTATION POINTERS	512 WORDS
0x1C00-0x1FFE	CHN 3 SEGMENTATION POINTERS	512 WORDS

The DSP is heavily involved in supporting segmentation. Segmentation requires the DSP to monitor the progress of each waveform. Once a waveform has started a new segment, a flag is set in the engine hardware. This flag indicates to the DSP it is time to load the next segments starting and ending address pointers.

The DSP has until the end of the current waveform segment to update these registers. This update time places real-time processing constraints on the DSP. For this reason, it also places constraints on the user's use of segmentation.

Each of the users 'waveform segments' must be long enough for the DSP to have time to update the next segment's address pointers before that segment becomes active. If the segment is not long enough, the DSP will miss the intended new segment and instead, the last segment will be replayed. The faster the sample clock is running, the less time the DSP has to update the new pointers. This equates to a limit on the minimum number of samples each segment can have. This limit is 0x100.

There are four constraints when allocating space for segmentation pointer programming:

- The starting address pointer for channel zero must be installed at 0x1000.
- The starting address pointer for channel one must be installed at 0x1400.
- The starting address pointer for channel two must be installed at 0x1800.
- The starting address pointer for channel three must be installed at 0x1C00.

The pointers are programmed in eight-word segments:

FIRST WORD Starting address pointer MSB, points to an address in waveform data space. Only one bit used.

SECOND WORD Starting address pointer LSB, points to an address in waveform data space. All 16 bits used.

THIRD WORD Ending address pointer MSB, points to an address in waveform data space. Only one bit used.

FOURTH WORD Ending address pointer LSB, points to an address in waveform data space. All 16 bits used.

FIFTH WORD Link pointer MSB points to an address in segmentation data space (VME 0x1000-0x13FF for channel). This address is the next first byte of a starting address pointer. Only one bit used.

SIXTH WORD Link pointer LSB points to an address in segmentation data space (VME 0x1000-0x13FFh for channel). This address is the next first byte of a starting address pointer. All 16 bits used.

SEVENTH WORD RESERVED

EIGHTH WORD RESERVED

3.5 VME-4145 Commands

The VME-4145's DSP contains embedded firmware that controls a number of different board functions. The following sections describe each of these functions. The descriptions given cover a brief overview of the usage of the command, the registers which are utilized by each command, and how the registers are to be set up.

The commands are entered by the host computer. In most cases, after the commands are processed, the Command Response Register (CRR) gives a code that will indicate whether the command has been accepted or rejected. In some cases, other registers are also used to pass additional information before and after the command execution. See the individual sections for details.

Table 3-30 below provides a list of commands the host computer may send to the VME-4145's onboard DSP processor. Hexcodes which are not defined are reserved for factory use and should not be used.

Table 3-30 Index of Commands on the VME-4145

Hexcode	Name	Hexcode	Name
0x0000	Null	0x0014	Load Clock Mux.
0x0001	Load General Configuration	0x0019	Enable Segmentation
0x0003	Load Channel Configuration	0x001A	Disable Segmentation
0x0004	Load Idle Time Command	0x001B	Report DSP Hardware Status
0x0005	Initialize CHN Eng./ Halt Wave Generation	0x001C	Reserved
0x0006	Load Full Waveform Table without Gain/ Offset Autocal.	0x001D	Software Reset
0x0007	Software Trigger CHN Command	0x0020	Calibrate Channels Command
0x0009	Load Sample Rate	0x0021	Reserved
0x000A	Generate Automatic Square Wave	0x0022	Reserved
0x000D	Ping DSP	0x0023	Load Full Waveform Table With Gain/Offset Autocal.
0x000E	Built-In-Test Command	0x0024	Reserved
0x0010	Reserved	0x0025	Download Cal. Data from E^2PROM to VME RAM
0x0011	Report EPLD Eng. Status Command	0x0026	Upload Cal. Data from VME RAM to E^2PROM
0x0012	Load Repeat Count	0x0027	DSP/ADC/DAC Self-Test Command
0x0013	Load Starting/Ending Add. Pointers	Reserved	Reserved

3.5.1 Command: Null

VCR Command Code: 0x0000

Description:

This is a no action command flag. During idle periods, the VCR is normally in this state. The VME user must place a nonzero value into this register to invoke the DSP's command processor. After the DSP processes a given command (VCR not equal to zero), it will clear the VCR back to zero. The clearing of this register back to zero is the flag to the VME user that his command was executed. The user may then read the CRR for a command execution return code.

Input Registers:

NONE

Output Registers:

CRR

3.5.2 Command: Load General Configuration

VCR Command Code: 0x0001

Description:

This command loads configuration data of a general nature. This register affects all four channels of operation or the general operation of the hardware.

Input Registers:

GCR

Output Registers:

CRR

NOTE

The GCR must be programmed before any waveforms are loaded. The GCR contains a bit defining user data as two's complement (default) or binary.

3.5.3 Command: Load Channel Configuration

VCR Command Code: 0x0003

Description: Offline Status

This is a command to load the individual operating mode for each engine. The channel's online output range and trigger configuration are programmed through the engine's mode select register. This register may be programmed while a channel is running.

Input Registers:

- CHN:Points to which channel's configuration to update
- CCR0:Channel zero configuration register
- CCR1:Channel one configuration register
- CCR2:Channel two configuration register
- CCR3:Channel three configuration register

Output Registers:

CRR

See *Channel Configuration Registers (CCR0 - CCR3)* for a definition of CCR0-CCR3 registers.

NOTE

The channel configuration must be set prior to loading a waveform for calibration. This will instruct the DSP which coefficient it should apply during calibration.

3.5.4 Command: Load Idle Time

VCR Command Code: 0x0004

Description:

This command is used to load the idle time counter. This command uses a 24-bit counter that sets the amount of delay (25ns units for a 40MHz master clock) between the end of the initial waveform period and the start of the next waveform period. A value of zero will result in no delay between the waveform cycles. This register may be programmed while a channel is running.

Input Registers:

CHN:Points to which channel's idle time count to update

IH0, 1, 2, 3

IL0, 1, 2, 3

Output Registers: CRR

3.5.5 Command: Initialize Channel Engine or Halt Waveform Generation

VCR Command Code: 0x0005

Description:

This is a command to initialize the waveform engine to a known starting point. Bits CSA2 - CSA0 of the selected channel status register (CSR0 - CSR3) should read zero after this command has been processed. This command can be used to HALT a waveform that is currently being generated. Once this command is issued, the waveform will immediately halt its output staying at the last value sent out to the DAC. The output may then be taken offline if desired. Please refer to the state diagram, Figure 3-1 on page 60. The Halt command causes the engine to go back to the '000' binary state. Issuing another trigger command is not possible at this time, since the unit is not in the **ARMed,** '011' state. To move from the '000' to the '011' state, the user must first issue a load start address command, 0x13, which puts the engine in the '001' load state. It is not necessary to reload the waveform or issue a 0x06 or 0x23 command. Then a 0x12, load repeat count must be issued with a nonzero value to move from the load to the '011' **ARMed waiting for trigger** state. Finally, you may reissue a trigger event to run the already loaded waveform.

An alternate technique to halt the waveform is to rewrite the repeat cycle register(s) RCR0-3 with a one. This will cause the engine to complete the current cycles output and halt on the last value in your waveform table. This will take the

engine from the '111,' **Wave** state to the '100,' **Done** state. To restart the wave generation, you can issue another 0x12, load repeat count with a nonzero value to move back to the '011,' **ARMed waiting for trigger** state. To restart another waveform, issue another trigger event.

Input Registers: CHN: Points to which channel to initialize

Output Registers: CRR

3.5.6 Command: Load Full Waveform Table without Gain/Offset Autocalibration

VCR Command Code: 0x0006

Description:

This command causes the DSP to load the user's waveform data from the shared VME RAM through the DSP into the specified output waveform memory. The DSP will upload the values into the actual waveform RAM. Once the data has been uploaded, the user must also reprogram the sample clock, the clock multiplexer, and provide an external or internal trigger.

Any size word, from 1 word to 56 (57,344) Kwords, may be loaded. If a waveform is larger than 56 K, the user must load the first 56 K and then follow with the remainder in a second load up to a total of 64 Kwords (65,536).

Input Registers:

- CHN: Points to channel to load.
- GCR: General Configuration Register
- SARH0, 1, 2, 3 SARL0, 1, 2, 3
- EARH0, 1, 2, 3 EARL0, 1, 2, 3
- DSRH0, 1, 2, 3 DSRL0, 1, 2, 3
- DERH0, 1, 2, 3 DERL0, 1, 2, 3
- CCR0, 1, 2, 3

The destination registers are 17-bit values. These registers form a pointer in waveform memory address space to the starting location in which the user wishes to load. The 17th bit destination address is formed by 1 bit from DSRH and 16 bits from DSRL.

The destination register does not have to match the starting address value. This is due to the fact that a user may have to perform two separate loads in order to load a buffer larger than 56 K. The remaining upper 8 K will be loaded at a starting destination after the 56 K ending point. Also, a user may wish to only update a single waveform segment in a segmented implementation.

The starting and ending address pointers are referring to the VME-shared RAM addresses. The starting address must be above the VME-4145's base address + 0x3FFEh. The ending address must be greater than the starting address value. The waveform size will be the ending address minus the starting address.

Example: If the base address were at 0x80000000, the first available waveform word would be at 0x80004000.

Output Registers: CRR

3.5.7 Command: Software Trigger Channel Command

VCR Command Code: 0x0007

Description:

This command is used to trigger a waveform generation by software means. Once a waveform is loaded, the unit will stay in the armed state until the software trigger command signals the start of the waveform.

Input Registers:

CHN: Points to which channel to trigger

Output Registers:

CRR

3.5.8 Command: Load Sample Rate

VCR Command Code: 0x0009

Description:

This command causes the DSP to reload the sample rate clock.

Input Registers:

- CHN: Points to channel to update.
- SRH0, 1, 2, 3 SRL0, 1, 2, 3

NOTE

Only one set of registers 0, 1, 2, or 3 is loaded at one time.

Output Registers:

CRR

3.5.9 Command: Generate Automatic Square Wave

VCR Command Code: 0x000A

Description:

This command allows the DSP to automatically set up the VME-4145 and generate a standard square wave on the DAC channel pointed to by the CHN register.

This allows the user to quickly generate a standard waveform. By issuing this single command, the channels configuration, clock, waveform address pointers, and waveform data are preset for the user. The same routines invoked by the user are used by the DSP to set up the unit. This provides a hardware/firmware confidence test. The user may also customize the waveform by setting a new clock frequency, idle time, repeat count, or range configuration while the waveform is being generated. If the user has a question about the correct value of which to set one of the registers, he may examine the register space of the VME-4145 after executing this command (for example: address starting and ending pointers).

3.5.10 Command: Ping DSP

VCR Command Code: 0x000D

Description:

This command is used to test the VCR command and CRR response mail box command mechanism of the VME host and DSP onboard processor. The user issues the command and the DSP will respond in the CRR with the value 0xBEEF. This ensures the user that the DSP is addressed correctly in memory space, that it is ready to accept commands to execute and can issue correct response codes. This command can be issued anytime after the DSP has gone completely through its power-on cycle or has finished a previously issued command.

3.5.11 Command: Built-In-Test Command

VCR Command Code: 0x000E

Description:

This command is used as a confidence test of the onboard 20-bit ADC and the four DAC output channels.

Setting the CMD_DATA_LSW register with a 0x0 and issuing the BIT command will test the ADC and its programmable attenuator. Multiple data points will be averaged at each configuration using analog ground and the +2.5, 5, and 10 V onboard voltage references. The resultant values will be stored back in VME waveform memory starting at the 0x4000 location (first word of VME waveform RAM).

Table 3-31 Built-in-Test Command Reference Values

BASE+	TYPICAL VALUE	REF.	GAIN	ADC INPUT	FIXED ADC (0.909)	ADC FIXED INPUT	BIPOLAR	UNIPOLAR
ATTENUATOR x1								
0x4000	h[0]=0x0002	GND	x1	0V	0.909	0V	0x7FFFF	0x00000
0x4002	l[0]=0xDC20	GND	x1	0V	0.909	0V	0x7FFFF	0x00000
0x4004	h[1]=0x7F63	2.5V	x1	2.5V	0.909	2.273V	0XF4609	0xE8C14
0x4006	l[1]=0xBFA0	2.5V	x1	2.5V	0.909	2.273V	0XF4609	0xE8C14
ATTENUATOR x1/2								
0x4008	h[2]=0x0002	GND	x0.5	0V	0.909	0V	0x7FFFF	0x00000
0x400A	l[2]=0x13A0	GND	x0.5	0V	0.909	0V	0x7FFFF	0x00000
0x400C	h[3]=0x3AE0	2.5V	x0.5	1.25V	0.909	1.136V	0xBA29B	0x74538
0x400E	l[3]=0x3300	2.5V	x0.5	1.25V	0.909	1.136V	0xBA29B	0x74538
0x4010	h[4]=0x75BC	5.0V	x0.5	2.5V	0.909	2.273V	0XF4609	0xE8C14
0x4012	l[4]=0x2EC0	5.0V	x0.5	2.5V	0.909	2.273V	0XF4609	0xE8C14
ATTENUATOR x1/4								
0x4014	h[5]=0x0006	GND	x0.25	0V	0.909	0V	0x7FFFF	0x00000
0x4016	l[5]=0x0EE0	GND	x0.25	0V	0.909	0V	0x7FFFF	0x00000
0x4018	h[6]=0x1D68	2.5V	x0.25	0.6125V	0.909	0.568V	0x9D14D	0x3A24C

Table 3-31 Built-in-Test Command Reference Values (Continued)

BASE+	TYPICAL VALUE	REF.	GAIN	ADC INPUT	FIXED ADC (0.909)	ADC FIXED INPUT	BIPOLAR	UNIPOLAR
0x401A	l[6]=0x2360	2.5V	x0.25	0.6125V	0.909	0.568V	0x9D14D	0x3A24C
0x401C	h[7]=0x3AC9	5.0V	x0.25	1.25V	0.909	1.136V	0xBA29B	0x74538
0x401E	l[7]=0x4FC0	5.0V	x0.25	1.25V	0.909	1.136V	0xBA29B	0x74538
0x4020	h[8]=0x7591	10.0V	x0.25	2.5V	0.909	2.273V	0XF4609	0xE8C14
0x4022	l[8]=0x6EE0	10.0V	x0.25	2.5V	0.909	2.273V	0XF4609	0xE8C14

The DSP left justifies the 20-bit ADC data into a 32-bit number using the format in below.

Table 3-32 DSP Output Data Format

D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
20-bit ADC Data Mapped into 32-bit Number																															
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0												
9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												

This number is a two's complement number and must be exclusively OR'ed with 0x8000 0000 to obtain the binary value.

Finally, there is a fixed attenuator in front of the ADC. It is set to a fixed 0.909 value. Therefore, if the ADC is reading the +2.5 reference at a x1 gain, the actual reading returned will be about +2.5 * 0.909 or +2.27 VDC.

Setting the CMD_DATA_LSW register with a 0x1 and issuing the BIT command will loopback the output connector voltage of the DAC channel pointed to by the CHN register into the ADC. The 20-bit value will be returned in the two registers BIT_ADC_HI and BIT_ADC_LO. The format is also two's complement 20-bit left-justified. The DAC output must not be changing while this command is running. There are a number of techniques to use to guarantee a stable DAC voltage while the ADC is converting the value. A constant waveform of a single value may be running. A waveform with a cycle count of one may be triggered, where the last table value is the one desired to measure. Finally, a waveform that is software single step clocked may be measured at each step.

Maximum and minimum tolerances within ± some number of bits should be expected in the answers returned. The answers should be used to gauge general circuit operation.

Table 3-33 ADC Attenuator Input Port Definitions

PORT	DEFINITION
Port 0	Channel 0
Port 1	Channel 1
Port 2	Channel 2
Port 3	Channel 3
Port 4	Ground
Port 5	+10VDC
Port 6	+5VDC
Port 7	+2.5VDC

Input Registers:

CHN:Points to desired DAC channel

CMD_DATA_LSW:0x0 = ADC, Attenuator and voltage reference test
0x1 = DAC, Loopback to ADC test

Output Registers:

- CRR
- BIT_ADC_HI
- BIT_ADC_LO

3.5.12 Command: Report EPLD Engine Status Command

VCR Command Code: 0x0011

Description:

This command is used to instruct the DSP to update the DAC engine status register in local VME register space.

Input Registers:

CHN: Points to which channel's status is being requested

Output Registers:

CRR

CSR0, CSR1, CSR2, or CSR3: Updated status register value is returned in the registers corresponding to the channel entered in the CHN register.

3.5.13 Command: Load Repeat Count

VCR Command Code: 0x0012

Description:

This command sets up the number of times the waveform will repeat before halting. The last value in the waveform will be held at the DAC output during the halted state.

Input Registers:

- CHN: Points to which channel's count is being programmed
- RCR0
- RCR1
- RCR2
- RCR3

Depending on what CHN is set to, only one of these registers is read and the associated DAC engine controller updated. A 9-bit value from zero to 511, in increments of one, can be programmed. **A value of 511 instructs the engine to generate a continuous waveform until the engine is reinitialized**.


NOTE

This register may be programmed while a channel is running.

Output Registers:

CRR


NOTE

This command must be issued after loading a waveform. The engine's state machine is forced from the 'load' to the 'arm' (waiting for the trigger) by writing a nonzero value to the RCR.

3.5.14 Command: Load Starting and Ending Address Pointers

VCR Command Code: 0x0013

Description:

Used to load the waveform buffer memory pointers.

This command does not cause the loading of VME RAM waveforms to DAC waveform space. See the load waveform command for performing this function. This command is used to reload memory pointers for waveform data which has previously been loaded. It is also used to reset the pointers when going from the segmentation mode back to the standard single waveform mode (assuming the user's wishes to utilize the preexisting waveforms used for segmentation).


NOTE

May be issued while a waveform is running.

Input Registers:

- CHN
- DSRH0,1,2,3
- DSRL0,1,2,3
- DERH0,1,2,3
- DERL0,1,2,3

Only one set of these address pointer register sets (0, 1, 2, or 3) is loaded at a time depending on the value of the CHN register.

In the examples on the next page, the first (channel zero) register set is shown. The other three register sets are identical in function.

Output Registers:

CRR

3.5.15 Command: Load Clock Multiplexer

VCR Command Code: 0x0014

Description:

This command controls the selection of the sample clock for channels 0-3. The channel may use its own internal dedicated clock, an external clock provided by the user or a common clock. The common clock is channel zero's clock source (internal or external source). Selecting a common clock allows multiple channels to have synchronized outputs. The user may also directly control the logic state of the clock from software. In this way, the user may generate sample clocks using the VME host processor. This is very useful for extremely long sample periods. When using the software-controlled clock (FORCE CLOCK LOW/HIGH), please note that it takes three clock cycles for the engine to register a trigger event. After these three clocks, the next rising edge clock will send out the first value in your waveform table. Therefore, you should follow the example below:

1. Force clock LOW
2. Issue Software Trigger Command
3. Issue Three Sets of: Force clock HIGH/LOW
4. Loop Issuing one Set of: Force clock HIGH/LOW for each sample

Input Registers:

CMR: The CMR (Clock MUX Register) is loaded with a multiplexer control value as described in Table 3-22 on page 65. The DSP will use this value to program the actual clock multiplexer device which is not directly accessible to the user.

Output Registers:

CRR

Example: Setting CMR to 0x0000 will program all four channels to use their own internal timebases.

3.5.16 Command: Enable Segmentation

VCR Command Code: 0x0019

Description:

This is a command used to turn on the segmented waveform mode of operation. The user must reload the SARs and EARs before generating any further waveforms (VCR command 0x0013).

Input Registers:

CHN: Points to channel being enabled.

Output Registers:

CRR

3.5.17 Command: Disable Segmentation

VCR Command Code: 0x001A

Description:

This is a command used to turn off the segmented waveform mode of operation. The user must reload the SARs and EARs before generating any further waveforms (VCR command 0x0013).

Input Registers:

CHN:Points to channel being disabled.

Output Registers: CRR

3.5.18 Command: Report DSP Hardware Status

VCR Command Code: 0x001B

Description:

This is a command to read the DSP's external hardware status register. This register contains bits that define the operation of the hardware. The value returned should be ANDed with 0x00FC to remove the factory use only bits.

Input Registers:

NONE

Output Registers:

- CDR:The CDR is loaded with the DSP status register value.
- CMD_DATA_LSW:This register is loaded with the current contents of the hardware control register. This is used when a user wishes to determine the state of the hardware output control register of the DSP.
- CRR

Table 3-34 DSP Hardware Status Register Bit Map

DSP HARDWARE STATUS REGISTER							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
Reserved		ADC RDY		Reserved			WDOG TimeOut*
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
UIOC*	4CHN*	TRIG3	TRIG2	TRIG1	TRIG0	Reserved	Reserved

*Denotes an active low signal

Hardware Status Register Bit Definitions

Bits [15:14] **Reserved:** These bits are reserved for factory use only.

Bit 13 **ADC RDY*:** This bit is reserved for factory use only.

Bit 8 **WDOG TIMEOUT*:** Active low indicates a watchdog timeout has occurred. Factory use only.

Bit 7 **UIOC*:** When read this bit indicates that the VME-4145 is used in a system with a UIOC controller.

Bit 6 **4CHN*:** Active low indicates the mezzanine card is installed.

Bits [5:2] **TRIG[3..0]:** Active high indicates a trigger condition has been requested at the front panel. This is a nonlatched real-time status.

Bit 1 **Reserved:** This bit is reserved.

Bit 0 **ADC SDATA:** This bit is reserved for factory use only.

Table 3-35 DSP Hardware Control Register Bit Map

DSP HARDWARE STATUS REGISTER							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08
FC2	FC1	MA5	MA4	MA3	MA2	MA1	MA0
Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00
ADCCS*	ADCUP*	ADCCAL	ADCSC2	ADCSC1	CONFIG*	ECLK	EDATA

* Denotes an active low signal

DSP Hardware Control Bit Definitions

Bit 15 **FC2:** Controls the onboard green LED, D18. A logical 0 = LED on, a logical 1 = LED off.

Bit 14 **FC1:** Controls the onboard green LED, D19. A logical 0 = LED on, a logical 1 = LED off.

All of the following signals are factory reserved signals and should not be changed from their present state: **MA0-5**, **ADCCS***, **ADCUP***, **ADCCAL**, **ADCSC1**, **ADCSC2**, **CONFIG***, **ECLK**, **EDATA**

3.5.19 Command: Software Reset

VCR Command Code: 0x001D

Description:

This command causes the DSP to reboot and run through its power on reinitialization and system test. During reset, the unit will turn off RAM access from the VME. During this time, the unit will respond at all addresses with the unit's board ID (0x3C00). This command does not set the response code.

Input Registers:

NONE

Output Registers:

NONE

3.5.20 Command: Calibrate Channels

VCR Command Code: 0x0020

Description:

This command is used to initiate a recalibration of the board. During this time, the unit will respond at all addresses with the unit's board ID (0x3C00). The new calibration data will be contained within the VME RAM area. If the user wishes to

make these calibration values permanent, the user must issue the Upload to E^2PROM, 0x26 command, (See *Command: Download CAL Data from E^2PROM to VME RAM*).

Input Registers:

NONE

Output Registers:

CRR

3.5.21 Command: Load Full Waveform Table with Gain/Offset Autocalibration

VCR Command Code: 0x0023

Description:

This command causes the DSP to load the user's waveform data from the shared VME RAM through the DSP into the specified output waveform memory. The DSP must perform autocalibration on all of the waveform data in place. Gain and offset corrections will be performed on every sample offline. Finally, the DSP will upload the calibrated values into the actual waveform RAM. The VME host must wait while calibration corrections and loading occur. Due to this fact, a waveform may not be loaded in real time. If a user wishes to achieve faster loading, the manual calibration mode may be invoked. This will bypass the DSP calibration step. During this DSP busy period, the unit will respond at all addresses with the unit's board ID (0x3C00). Once the data has been uploaded, the user must also reprogram the sample clock, the clock multiplexer, and provide an external or internal trigger.

Any size word, from 1 word to 56 (57,344) Kwords, may be loaded. If a waveform is larger than 56 K, the user must load the waveform pieces.

Input Registers:

- GCR (2)
- CHN: Points to channel to load
- CCR0, 1, 2, 3 (1)
- SARH0, 1, 2, 3 SARL0, 1, 2, 3
- EARH0, 1, 2, 3 EARL0, 1, 2, 3
- DSRH0, 1, 2, 3 DSRL0, 1, 2, 3
- DERH0, 1, 2, 3 DERL0, 1, 2, 3

The destination register is a pointer in waveform memory address space to the starting location in which the user wishes to load.

The destination register does not have to match the starting address value. This is due to the fact that a user may have to perform two separate loads in order to load a buffer larger than 56 K. The remaining upper 8 K will be loaded at a starting destination after the 56 K ending point. Also, a user may wish to only update a single waveform segment in a segmented implementation.

The starting and ending address pointers are referring to the VME shared RAM addresses. The starting address must be above the VME-4145's base address +

0x3FFEh. The ending address must be greater than the starting address value. The waveform size will be the ending address minus the starting address.

Example: If the base address were at 0x80000000, the first available waveform word would be at 0x80004000.

NOTE

The channel configuration registers must be set prior to issuing this command. The DSP must know the desired output range so that it can apply the correct calibration coefficients.

NOTE

If GCR bit 0 equals one, the DSP will replace the user's uncalibrated data with calibrated data in the VME waveform buffers. This allows the user to see the actual values presented to the DAC. If GCR bit 0 equals a zero, the DSP will not corrupt the local user VME data.

Output Registers: CRR

0x1 or 0x4: Waveform loaded successfully.

0x5: Waveform loaded with clipping. The DSP had underflow or overflow occur while applying correction coefficients to the user's data. Values will be clipped to minimum or maximum output range. Recalibration of the unit is suggested.

3.5.22 Command: Download CAL Data from E²PROM to VME RAM

VCR Command Code: 0x0025

Description:

This command causes the DSP to download the E^{2}PROM saved calibration gain and offset values to be loaded into VME RAM, which is accessible to the user.

Input Registers: NONE

Output Registers: CRR

3.5.23 Command: Upload CAL Data from VME RAM to E²PROM

VCR Command Code: 0x0026

Description:

This command causes the DSP to upload the calibration gain and offset values loaded in the user's VME RAM space to the E^{2}PROM for permanent storage.

Input Registers: NONE

Output Registers: CRR

3.5.24 Command: DSP/ADC/DAC Self-Test

VCR Command Code: 0x0027

Description:

This command causes the DSP to run a set of tests which checks the DSP, the VME RAM, and the Waveform RAM.

Input Registers: NONE

Output Registers: CRR and SSR

Maintenance

If a GE Fanuc Intelligent Platforms product malfunctions, please verify the following:

1. Software version resident on the product
2. System configuration
3. Electrical connections
4. Jumper or configuration options
5. Boards are fully inserted into their proper connector location
6. Connector pins are clean and free from contamination
7. No components or adjacent boards were disturbed when inserting or removing the board from the chassis
8. Quality of cables and I/O connections

If products must be returned, contact GE Fanuc Intelligent Platforms for a Return Material Authorization (RMA) Number. **This RMA Number must be obtained prior to any return.** The RMA is available at rma@gefanuc.com.

GE Fanuc Intelligent Platforms Customer Care is available at: 1-800-GEFANUC (or 1-800-433-2682), 1-780-401-7700. Or, E-mail us at support.embeddedsystems@gefanuc.com.

Maintenance Prints

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

A • Appendix A: Sample Codes

This sample code is the common header file for the three following example codes. This file defines the structures of all of the registers used by the VME-4145, the segmentation programming areas, and the VME waveform RAM.

A.1 4145.h

```
/* VME-4145 6U analog function generator board
4145.h
  Header file for user.c examples
  02/06/08 Bhaskar.Challa Udaya GE Fanuc Intelligent Platforms
*/

/* define a structure with sequential registers matching 4145 */
struct vmivme_4145_registers {
/* addresses relative to  */
/* DSP ADDR ( double for VME) */
unsigned short bid; /* 0x00 Board ID Register */
unsigned short chn;/* 0x02 channel pointer number*/
unsigned short vcr;/* 0x04 vme command register*/
unsigned short cdr;/* 0x06 configuration data register*/
unsigned short crr;/* 0x08 command response register*/
unsigned short gcr;/* 0x0A general configuration register*/
unsigned short ssr;/* 0x0C self-test status register*/
unsigned short rst;/* 0x12 software reset register*/
unsigned short ewr;/* 0x14 e2prom writes register*/
unsigned short cmd_data_msb;/* 0x16 */
unsigned short cmd_data_lsb;/* 0x18 */
unsigned short bit_adc_hi; /* 0x1A Board Ready Register in RAM */
unsigned short bit_adc_lo;/* 0x1C Board Ready Register in RAM */
unsigned short ffr;/* 0x1E firmware revision register*/
unsigned short cs0;/* 0x20 chn 0 status register*/
unsigned short cs1;/* 0x22 chn 1 status register*/
unsigned short cs2;/* 0x24 chn 2 status register*/
unsigned short cs3;/* 0x26 chn 3 status register*/
unsigned short cc0;/* 0x28 chn 0 configuration register*/
unsigned short cc1;/* 0x2A chn 1 configuration register*/
unsigned short cc2;/* 0x2C chn 2 configuration register*/
unsigned short cc3;/* 0x2E chn 3 configuration register*/
unsigned short srh0;/* 0x30 sample rate MSB (4 bit high nibble)
register 0*/
unsigned short srl0;/* 0x32 sample rate LSB (16 bit low word)
register 0*/
unsigned short srh1;/* 0x34 sample rate MSB (4 bit high nibble)
register 1*/
unsigned short srl1;/* 0x36 sample rate LSB (16 bit low word)
register 1*/
unsigned short srh2;/* 0x38 sample rate MSB (4 bit high nibble)
register 2*/
unsigned short srl2;/* 0x3A sample rate LSB (16 bit low word)
register 2*/
unsigned short srh3;/* 0x3C sample rate MSB (4 bit high nibble)
register 3*/
```

```
unsigned short srl3;/* 0x3E sample rate LSB (16 bit low word)
register 3*/
unsigned short ih0;/* 0x40 chn 0 idle msb high 8 bits*/
unsigned short il0;/* 0x42 chn 0 idle lsb high 16 bits*/
unsigned short ih1;/* 0x44 chn 1 idle msb high 8 bits*/
unsigned short il1;/* 0x46 chn 1 idle lsb high 16 bits*/
unsigned short ih2;/* 0x48 chn 2 idle msb high 8 bits*/
unsigned short il2;/* 0x4A chn 2 idle lsb high 16 bits*/
unsigned short ih3;/* 0x4C chn 3 idle msb high 8 bits*/
unsigned short il3;/* 0x4E chn 3 idle lsb high 16 bits*/
unsigned short rcr0;/* 0x68 chn 0 repeat count register*/
unsigned short rcr1;/* 0x6A chn 1 repeat count register*/
unsigned short rcr2;/* 0x6C chn 2 repeat count register*/
unsigned short rcr3;/* 0x6E chn 3 repeat count register*/
unsigned short sarh0;/* 0x80 chn 0 VME start address MSB (1bit) */
unsigned short sarl0;/* 0x82 chn 0 VME start address LSB (16bit) */
unsigned short earh0;/* 0x84 chn 0 VME end address MSB (1bit) */
unsigned short earl0;/* 0x86 chn 0 VME end address LSB (16bit)*/
unsigned short dsarh0;/* 0x88 chn 0 wave ram destination start
address MSB (1bit) */
unsigned short dsarl0;/* 0x8A chn 0 wave ram  destination start
address LSB (16bit) */
unsigned short dearh0;/* 0x8C chn 0 wave ram  destination end addr
msb */
unsigned short dearl0;/* 0x8E chn 0 wave ram  destination end addr
lsb */
unsigned short sarh1;/* 0x90 chn 1 VME start address MSB (1bit) */
unsigned short sarl1;/* 0x92 chn 1 VME start address LSB (16bit) */
unsigned short earh1;/* 0x94 chn 1 VME end address MSB (1bit) */
unsigned short earl1;/* 0x96 chn 1 VME end address LSB (16bit) */
unsigned short dsarh1;/* 0x98 chn 1 wave ram  end address MSB
(1bit) */
unsigned short dsarl1;/* 0x9A chn 1 wave ram  end address LSB
(16bit) */
unsigned short dearh1;/* 0x9C chn 1 wave ram  destination end addr
msb */
unsigned short dearl1;/* 0x9E chn 1 wave ram  destination end addr
lsb */
unsigned short sarh2;/* 0xA0 chn 2 VME start address MSB (1bit) */
unsigned short sarl2;/* 0xA2 chn 2 VME start address LSB (16bit) */
unsigned short earh2;/* 0xA4 chn 2 VME end address MSB (1bit) */
unsigned short earl2;/* 0xA6 chn 2 VME end address LSB (16bit) */
unsigned short dsarh2;/* 0xA8 chn 2 wave ram start address MSB
(1bit) */
unsigned short dsarl2;/* 0xAA chn 2 wave ram start address LSB
(16bit) */
unsigned short dearh2;/* 0xAC chn 2 wave ram end address MSB
(1bit)*/
unsigned short dearl2;/* 0xAE chn 2 wave ram end address LSB
(16bit)*/
unsigned short sarh3;/* 0xB0 chn 3 VME start address MSB (1bit) */
unsigned short sarl3;/* 0xB2 chn 3 VME start address LSB (16bit) */
unsigned short earh3;/* 0xB4 chn 3 VME end address MSB (1bit) */
unsigned short earl3;/* 0xB6 chn 3 VME end address LSB (16bit) */
unsigned short dsarh3;/* 0xB8 chn 3 wave ram start address MSB
(1bit) */
unsigned short dsarl3;/* 0xBA chn 3 wave ram start address LSB
(16bit) */
```

```
unsigned short dearh3;/* 0xBC chn 3 wave ram end address MSB (1bit)
*/
unsigned short dearl3;/* 0xBE chn 3 wave ram end address LSB
(16bit) */
unsigned short cmr;/* 0xC0 clock mux register*/
};
typedef struct vmivme_4145_registers t_4145;

/* define four structures for segmentation programming */
struct seg_ram0 {
unsigned short smp0[1023]; /* 0x1000-0x13FE SEGMENTATION MEMORY
POINTER   CHN 0*/
};
typedef struct seg_ram0 seg0_4145;


struct seg_ram1 {
unsigned short smp0[1023]; /*  0x1400-0x17FE SEGMENTATION MEMORY
POINTER   CHN 1*/
};
typedef struct seg_ram1 seg1_4145;


struct seg_ram2 {
unsigned short smp0[1023]; /*  0x1800-0x1BFE SEGMENTATION MEMORY
POINTER   CHN 2*/
};
typedef struct seg_ram2 seg2_4145;


struct seg_ram3 {
unsigned short smp0[1023]; /*  0x1C00-0x1FFE SEGMENTATION MEMORY
POINTER   CHN 3*/
};
typedef struct seg_ram3 seg3_4145;


/* Define structure for first 32K portion of 56K waveform buffer*/
struct wave_ram {
unsigned short ram[32768]; /*  0x4000-0x13FFE */
};
typedef struct wave_ram r_4145;
```

A.2 user2.c

This code programs the VME-4145 to generate a simple square wave.

```
/*
VME-4145 4 channel arbitrary function generator
USER EXAMPLE FILE TO GENERATE A SIMPLE SQUARE WAVE
GE Fanuc Intelligent Platforms
Jun 02, 2008
Bhaskar.Challa Udaya
Written in cross code c
```

```
Related Files:

user2.c          General configuration & hardware setup
4145.h           Defines memory map using structures
*/

#include <stdio.h>
#include <stdlib.h>
#include <test.h>
#include <string.h>
#include <vt100.h>
#include  "4145.h"
#include <math.h>


/* assign vme physical addresses to data structures see 4145.h */
/* assume that the unit is configured for a base address of
0x80000000 */
/* (address jumpers: all but E12 1-2 installed, all on E2 installed
*/

t_4145 * uut      = (( t_4145 * )( 0x80000000 ));/* control&status
*/
r_4145 * wave     = (( r_4145 * )( 0x80004000 ));/* 56K waveform
area*/
seg0_4145 * seg0  = (( seg0_4145 * )( 0x80001000 ));/* segment
programming  chn 0 */
seg1_4145 * seg1  = (( seg1_4145 * )( 0x80001400 ));/* segment
programming  chn 1 */
seg2_4145 * seg2  = (( seg2_4145 * )( 0x80001800 ));/* segment
programming  chn 2 */
seg3_4145 * seg3  = (( seg3_4145 * )( 0x80001C00 ));/* segment
programming  chn 3 */

void wait_busy();

main() {

char csr_byte;
unsigned short chnpntr,j,cmd,c,c1,ncount,msr,high,low;
/* VME addresses telling DSP where you put the waveform in VME ram
*/
unsigned short starth,startl;
unsigned short endh,endl;
/* Addresses telling the DSP where you to move the waveform into
DAC ram */
unsigned short  startdesth,startdestl;
unsigned short enddesth,enddestl;

uut->chn=0   ;  /* setup DAC channel zero  */
cmd=0x1A;/*disable segmentation command */
uut->vcr=cmd;
wait_busy();/* loop waiting for dsp to finish  command */
cmd=0x05;/*init dac command */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */
uut->gcr=0x0E;/* send gcr contents */
cmd=0x01;/* load gcr command */
```

```
uut->vcr=cmd;/* send command to load*/
wait_busy();/* wait for command to execute */

/* load a waveform into 4145 VME ram */

ncount=0;
c=0xffff;
printf("Square Waveform, 8K samples\r\n");
for(j=0;j<=0x1000;j++){
wave->ram[ncount]=c;
ncount++;
}
c=0x0000;
for(j=0;j<=0x1000;j++){
wave->ram[ncount]=c;
ncount++;
}

ncount--;/* adjust count for end of for loop */
starth = endh = startdesth = 0; /* if waveform <16K words msb=0 */
enddesth = enddestl = 0;        /* dsp will calculate these */
startl = 0x4000;/* start at 0x80004000, start of waveram*/
startdestl  = 0;
/* start of page 1 (0x4000) + double wave sample count(for vmeaddr)
*/
endl   = startl + 2 * ncount;

uut->dsarh0=startdesth;
uut->dsarl0=startdestl;
uut->dearh0=enddesth;
uut->dearl0=enddestl;
uut->sarh0=starth;
uut->sarl0=startl;
uut->earh0=endh;
uut->earl0=endl;

printf("LOADED Waveform address pointers \r\n");
printf("VME RAM starthi = %xh, startlo = %xh \r\n",starth,startl);
printf("VME RAM endhi = %xh, endlo = %xh \r\n",endh,endl);
printf("DAC RAM startdesthi = %xh, startdestlo = %xh
\r\n",startdesth,startdestl);

cmd=0x6; /*(load waveform without autocal ) */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

msr = 0x106F;/* bipolar, +/-2.5, software trigger */
uut->cc0=msr;
cmd=0x03;/* load channel configuration */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

uut->srh0=0x0;
uut->srl0=0x0F;
cmd=0x09;
uut->vcr=cmd;/* load sample rate command */
wait_busy();/* wait for command to execute */
```

```
uut->ih0=0x0;
uut->il0=0x0;
cmd=0x04;
uut->vcr=cmd;/* load idle timer command */
wait_busy();/* wait for command to execute */

ncount = 511;/* continuous waveform */
uut->rcr0=ncount;
cmd=0x12;
uut->vcr=cmd;/* load cycle count command */
wait_busy();/* wait for command to execute */

c=0;
uut->cmr=c;
cmd=0x14;
uut->vcr=cmd;/* load clock mux register */
wait_busy();/* wait for command to execute */

cmd=0x7;
uut->vcr=cmd;/* software trigger the waveform */
wait_busy();/* wait for command to execute */

cmd=0x11;
uut->vcr=cmd;/* ask dsp for epld status code */
wait_busy();/* wait for command to execute */

c=uut->cs0;
printf("epld status channel 0=       %4x \r\n",c);

c&=0x7;
switch(c){
          case 0x00:
              printf("     initialized state \r\n");
          break;
          case 0x01:
              printf("     load state \r\n");
          break;
          case 0x03:
              printf("    armed waiting for trigger state \r\n");
          break;
          case 0x07:
              printf("     generating waveform state \r\n");
          break;
          case 0x06:
              printf("idle state \r\n");
          break;
          case 0x04:
          printf("wave generation done  state \r\n");
          break;
}
for (;;){ }/* endless loop */
}

/* Wait for dsp to complete it's given command */
/* DSP will set board id 0x3C00  for any reads from 4145 until its
done */
```

```
void wait_busy(){
unsigned short c;
printf("\r\nWaiting for VCR register != 0 (hit any key to
abort)\r\n");
while(!kbhit() & (c=uut->vcr != 0) ) {};/* wait for command to be
cleared*/
return;
   }
```

A.3 user3.c

```
/*
VME-4145 4 channel arbitrary function generator
USER EXAMPLE FILE TO IMPLEMENT SEGMENTATION
GE Fanuc Intelligent Platforms
Jun 02, 2008
Bhaskar.Challa Udaya
Written in cross code c
Related Files:

user3.c General configuration & hardware setup
4145.h  Defines memory map using structures
*/
#include <stdio.h>
#include <stdlib.h>
#include <test.h>
#include <string.h>
#include <vt100.h>
#include  "4145.h"
#include <math.h>


/* assign vme physical addresses to data structures see 4145.h */
/* assume that the unit is configured for a base address of
0x80000000 */
/* (address jumpers: all but E12 1-2 installed, all on E2 installed
*/

t_4145 * uut      = (( t_4145 * )( 0x80000000 ));/* control&status
*/
r_4145 * wave     = (( r_4145 * )( 0x80004000 ));/* 56K waveform
area*/
seg0_4145 * seg0  = (( seg0_4145 * )( 0x80001000 ));/* segment
programming  chn 0 */
seg1_4145 * seg1  = (( seg1_4145 * )( 0x80001400 ));/* segment
programming  chn 1 */
seg2_4145 * seg2  = (( seg2_4145 * )( 0x80001800 ));/* segment
programming  chn 2 */
seg3_4145 * seg3  = (( seg3_4145 * )( 0x80001C00 ));/* segment
programming  chn 3 */

void wait_busy();
void hit_key();

main() {

charcsr_byte;
intwave_type;
```

```
unsigned short chnpntr,j,cmd,c,c1,ncount,msr,high,low,xx;
/* VME addresses telling DSP where you put the waveform in VME ram
*/
unsigned short starth,startl;
unsigned short endh,endl;
/* Addresses telling the DSP where you to move the waveform into
DAC ram */
unsigned short  startdesth,startdestl;
unsigned short enddesth,enddestl;
char    cb[80];/* keyboard general purpose input buffer */

printf("Load Segmented Waveforms   \r\n");

/* initialize hardware */
uut->chn=0   ;  /* setup DAC channel zero  */
cmd=0x1A;/*disable segmentation command */
uut->vcr=cmd;
wait_busy();/* loop waiting for dsp to finish  command */
cmd=0x05;/*init dac command */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */
uut->gcr=0x0E;/* send gcr contents */
cmd=0x01;/* load gcr command */
uut->vcr=cmd;/* send command to load*/
wait_busy();/* wait for command to execute */

uut->srh0=0x0;
uut->srl0=0x0F;
cmd=0x09;
uut->vcr=cmd;/* load sample rate command */
wait_busy();/* wait for command to execute */

uut->ih0=0x0;
uut->il0=0x0;
cmd=0x04;
uut->vcr=cmd;/* load idle timer command */
wait_busy();/* wait for command to execute */

/*  load two segmented waveforms                              */
/*  #1: load an 8K sample triangle in the bottom of memory   */
/*  #2: load an 8K sample square at the middle of the memory */


xx=0;
printf("Triangle Waveform 8k: 80004000-80007FFE samples
%i\r\n",chnpntr);
c=0;
       for(j=0;j<0x1000;j++) {
       wave->ram[xx]=c;
       c+=16;
       xx++;
       }
c=0xffff;
       for(j=0;j<0x1000;j++) {
       wave->ram[xx]=c;
       c-=16;
       xx++;
       }
```

```
starth = endh = startdesth = 0;
startl = 0x4000;
startdestl  = 0; /* bottom of 32K */
endl   = 0x7FFE;

/* set address pointers */
uut->dsarh0=startdesth;
uut->dsarl0=startdestl;
uut->dearh0=enddesth;
uut->dearl0=enddestl;
uut->sarh0=starth;
uut->sarl0=startl;
uut->earh0=endh;
uut->earl0=endl;

printf("AUTO LOADED Waveform #1 address pointers \r\n");
printf("VME starthi     = %5xh, startlo     = %5xh
\r\n",starth,startl);
printf("VME endhi       = %5xh, endlo       = %5xh
\r\n",endh,endl);
printf("ADC startdesthi = %5xh, startdestlo = %5xh
\r\n",startdesth,startdestl);

cmd=0x6; /*(load waveform without autocal ) */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

hit_key();

xx=0x0000;/* vme ram starting address */
printf("\r\n ");
printf("Square Waveform, 8K (32K offset)  80004000 - 80007FFE
samples\r\n");

for(j=0;j<0x1000;j++){
wave->ram[xx]=0xffff;
xx++;
}
for(j=0;j<0x1000;j++){
wave->ram[xx]=0x0000;
xx++;
}

starth = endh = startdesth = 0;
startl = 0x4000;
endl   = 0x7FFE;
startdesth  = 0x0001;
startdestl  = 0x0000;/* upper 1/2 of ram */

/* set address pointers */
uut->dsarh0=startdesth;
uut->dsarl0=startdestl;
uut->dearh0=enddesth;
uut->dearl0=enddestl;
uut->sarh0=starth;
uut->sarl0=startl;
uut->earh0=endh;
uut->earl0=endl;
```

```
printf("AUTO LOADED Waveform #2 address pointers \r\n");
printf("VME starthi     = %5xh, startlo     = %5xh
\r\n",starth,startl);
printf("VME endhi       = %5xh, endlo       = %5xh
\r\n",endh,endl);
printf("ADC startdesthi = %5xh, startdestlo = %5xh
\r\n",startdesth,startdestl);

cmd=0x6; /*(load waveform without autocal ) */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */


/* set engine pointers to run all of memory*/

startdesth=0;
startdestl=0;
enddesth=1;
enddestl=0xfffe;/* set a first time thru pointers */
/*starth=0;
startl=0;
endh=0;
endl=0;*/

/* set destination engine start, end pointers  */

uut->dsarh0=startdesth;
uut->dsarl0=startdestl;
uut->dearh0=enddesth;
uut->dearl0=enddestl;

cmd=0x13;/* reload DAC waveform starting and ending pointers */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

msr = 0x106F;/* bipolar, +/-2.5, software trigger */
uut->cc0=msr;
cmd=0x03;/* load channel configuration */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

/* writing rcr will clock engine from load to armed state */
ncount = 511;/* continuous waveform */
uut->rcr0=ncount;
cmd=0x12;
uut->vcr=cmd;/* load cycle count command */
wait_busy();/* wait for command to execute */

c=0;
uut->cmr=c;
cmd=0x14;
uut->vcr=cmd;/* load clock mux register */
wait_busy();/* wait for command to execute */

cmd=0x7;
uut->vcr=cmd;/* software trigger the waveform */
wait_busy();/* wait for command to execute */
```

```
/* now the waveform is running entire 64K which includes */
/* 8K triangle and 8K square plus remainder of memory junk */

for (;;){ /* endless loop */

printf("Select Segmented Waveform   \r\n");

printf("Enter  Type:\r\n");
printf("1) Triangle and Square  \r\n");
printf("2) Triangle only   \r\n");
printf("3) Square  only \r\n");

gets(cb);
sscanf(cb,"%i",&wave_type);

   if(wave_type==1) {


/* first wave is at start of wave memory for 8192 samples */
seg0->smp0[0] =0x0000;/* 0x1000 sar msb 1st wave */
seg0->smp0[1] =0x0000;/* 0x1002 sar lsb 1st wave */
seg0->smp0[2] =0x0000;/* 0x1004 ear msb 1st wave */
seg0->smp0[3] =0x3ffe;/* 0x1006 ear lsb 1st wave */
seg0->smp0[4] =0x0000;/* 0x1008 link msb 1st wave */
seg0->smp0[5] =0x1010;/* 0x100A link lsb 1st wave */
seg0->smp0[6] =0x0000;/* 0x100C waste it  */
seg0->smp0[7] =0x0000;/* 0x100E waste it  */

/* second wave is at 32K 1/2 way point for 8192 samples */

seg0->smp0[8] =0x0001;/* 0x1010 sar msb 2nd wave */
seg0->smp0[9] =0x0000;/* 0x1012 sar lsb 2nd wave */
seg0->smp0[10]=0x0001;/* 0x1014 ear msb 2nd wave */
seg0->smp0[11]=0x3ffe;/* 0x1016 ear lsb 2nd wave */
seg0->smp0[12]=0x0000;/* 0x1018 link msb 2nd wave */
seg0->smp0[13]=0x1000;  /* 0x101A link lsb 2nd wave (repeat wave
one ) */
seg0->smp0[14]=0x0000;/* 0x101C waste it */
seg0->smp0[15]=0x0000;/* 0x101E waste it */

}


   if(wave_type==2) {

/* first wave is at start of wave memory for 8192 samples */
seg0->smp0[0]=0x0000;/* sar msb 1st wave */
seg0->smp0[1]=0x0000;/* sar lsb 1st wave */
seg0->smp0[2]=0x0000;/* ear msb 1st wave */
seg0->smp0[3]=0x3fff;/* ear lsb 1st wave */
seg0->smp0[4]=0x0000;/* link msb 1st wave */
seg0->smp0[5]=0x1000;/* link lsb 1st wave */
seg0->smp0[6]=0x0000;/* waste it  */
seg0->smp0[7]=0x0000;/* waste it  */

   }

   if(wave_type==3) {
```

```
/* second wave is at 32K 1/2 way point for 8192 samples */
seg0->smp0[0]=0x0001;/* sar msb 2nd wave */
seg0->smp0[1]=0x0000;/* sar lsb 2nd wave */
seg0->smp0[2]=0x0001;/* ear msb 2nd wave */
seg0->smp0[3]=0x3fff;/* ear lsb 2nd wave */
seg0->smp0[4]=0x0000;/* link msb 2nd wave */
seg0->smp0[5]=0x1000;/* link lsb 2nd wave (repeat wave one ) */
seg0->smp0[6]=0x0000;/* waste it  */
seg0->smp0[7]=0x0000;/* waste it  */

    }


cmd=0x19;/* enable segmenation */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

  } /* endless loop */


}  /* end of main routine */

/* Wait for dsp to complete it's given command */
/* DSP will set board id 0x3C00  for any reads from 4145 until its
done */

void wait_busy(){
unsigned short c;
printf("\r\nWaiting for VCR register != 0 (hit any key to
abort)\r\n");
while(!kbhit() & (c=uut->vcr != 0) ) {};/* wait for command to be
cleared*/
return;
}

/*-------------------------------------------------------*/
void hit_key()
{
intc;

printf("Press any key to continue\r\n");
while(!kbhit());
return;
}
```

A.4 user4.c

```
/*
VME-4145 4 channel arbitrary function generator
USER EXAMPLE FILE TO GENERATE A SIMPLE 1Khz SINWAVE on channel 1
GE Fanuc Intelligent Platforms
Jun 02, 2008
Bhaskar.Challa Udaya
Written in cross code c
Related Files:
user4.c  General configuration & hardware setup
4145.h   Defines memory map using structures
```

```
*/
#include <stdio.h>
#include <stdlib.h>
#include <test.h>
#include <string.h>
#include <vt100.h>
#include  "4145.h"
#include <math.h>


/* assign vme physical addresses to data structures see 4145.h */
/* assume that the unit is configured for a base address of
0x80000000 */
/* (address jumpers: all but E12 1-2 installed, all on E2 installed
*/

t_4145 * uut = (( t_4145 * )( 0x80000000 ));/* control&status */
r_4145 * wave = (( r_4145 * )( 0x80004000 ));/* 56K waveform area*/
seg0_4145 * seg0= (( seg0_4145 * )( 0x80001000 ));/* segment
programming chn 0 */
seg1_4145 * seg1 = (( seg1_4145 * )( 0x80001400 ));/* segment
programming  chn 1 */
seg2_4145 * seg2  = (( seg2_4145 * )( 0x80001800 ));/* segment
programming  chn 2 */
seg3_4145 * seg3  = (( seg3_4145 * )( 0x80001C00 ));/* segment
programming  chn 3 */

void wait_busy();
void sin1();

unsigned short ncount=0;

main() {

char csr_byte;
extern unsigned short ncount;
unsigned short chnpntr,j,cmd,c,c1,msr,high,low;
/* VME addresses telling DSP where you put the waveform in VME ram
*/
unsigned short starth,startl;
unsigned short endh,endl;
/* Addresses telling the DSP where you to move the waveform into
DAC ram */
unsigned short startdesth,startdestl;
unsigned short enddesth,enddestl;

uut->chn=1   ;  /* setup DAC channel one  */

cmd=0x1A;/* disable segmentation command */
uut->vcr=cmd;
wait_busy();/* loop waiting for dsp to finish  command */

cmd=0x05;/*init dac command */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

uut->gcr=0x0E;/* send gcr contents */
cmd=0x01;/* load gcr command */
```

```
uut->vcr=cmd;/* send command to load*/
wait_busy();/* wait for command to execute */

/* load a waveform into 4145 VME ram */

sin1();/* load a sinwave */

ncount--;/* adjust count for end of for loop */
starth = endh = startdesth = 0; /* if waveform <16K words msb=0 */
enddesth = enddestl = 0;        /* dsp will calculate these */
startl = 0x4000;/* start at 0x80004000, start of waveram*/
startdestl  = 0;
/* start of page 1 (0x4000) + double wave sample count(for vmeaddr)
*/
endl   = startl + 2 * ncount;

uut->dsarh1=startdesth;
uut->dsarl1=startdestl;
uut->dearh1=enddesth;
uut->dearl1=enddestl;
uut->sarh1=starth;
uut->sarl1=startl;
uut->earh1=endh;
uut->earl1=endl;

printf("LOADED Waveform address pointers \r\n");
printf("VME RAM starthi = %xh, startlo = %xh \r\n",starth,startl);
printf("VME RAM endhi = %xh, endlo = %xh \r\n",endh,endl);
printf("DAC RAM startdesthi = %xh, startdestlo = %xh
\r\n",startdesth,startdestl);

cmd=0x6; /*(load waveform without autocal ) */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

msr = 0x106F;/* bipolar, +/-2.5, software trigger */
uut->cc1=msr;
cmd=0x03;/* load channel configuration */
uut->vcr=cmd;
wait_busy();/* wait for command to execute */

uut->srh1=0x0;
uut->srl1=0x0F;
cmd=0x09;
uut->vcr=cmd;/* load sample rate command */
wait_busy();/* wait for command to execute */


uut->ih1=0x0;
uut->il1=0x0;
cmd=0x04;
uut->vcr=cmd;/* load idle timer command */
wait_busy();/* wait for command to execute */

ncount = 511;/* continuous waveform */
uut->rcr1=ncount;
cmd=0x12;
uut->vcr=cmd;/* load cycle count command */
```

```
wait_busy();/* wait for command to execute */

c=0;
uut->cmr=c;
cmd=0x14;
uut->vcr=cmd;/* load clock mux register */
wait_busy();/* wait for command to execute */

cmd=0x7;
uut->vcr=cmd;/* software trigger the waveform */
wait_busy();/* wait for command to execute */

cmd=0x11;
uut->vcr=cmd;/* ask dsp for epld status code */
wait_busy();/* wait for command to execute */

c=uut->cs1;
c&=0x7;
printf("epld status channel 1=       %4x \r\n",c);
switch(c){
case 0x00:
printf("     initialized state \r\n");
break;
case 0x01:
printf("     load state \r\n");
break;
case 0x03:

printf("     armed waiting for trigger state \r\n");
break;
case 0x07:
printf("     generating waveform state \r\n");
break;
case 0x06:
printf("idle state \r\n");
break;
case 0x04:
printf("wave generation done  state \r\n");
break;

}
for (;;){ }/* endless loop */
/* Wait for dsp to complete it's given command */
/* DSP will set board id 0x3C00  for any reads from 4145 until its
done */

void wait_busy(){
unsigned short c;
printf("\r\nWaiting for VCR register != 0 (hit any key to
abort)\r\n");
while(!kbhit() & (c=uut->vcr != 0) ) {};/* wait for command to be
cleared*/
return;
}

voidsin1(){
float voffset,vpeak,span,ix,fdacstep,vlow,vhigh;
int nsamp;
```

```
double sinev,fj;
unsigned short isinev;
extern unsigned short ncount;

printf("Loading a SINWAVE\r\n");
vlow =-2.50;
vhigh=+2.50;
span = 5.0;/* +/- 2.5V Bipolar output has 5V span   */
nsamp = 2500;/* 1000Khz freq with 2.5 Msps output rate */
voffset=0.0;/* do not offset zero crossing point       */
printf("\r\nOffset voltage (for unipolar ranges) (0 for bipolar) ?
\r\n");
vpeak=2.5;/* maximum peak magnitude desired */
fdacstep= 6.28/nsamp;
printf("Range Low = %fV, Range high = %fV, Range span = %fV
\r\n",vlow,vhigh,span);
printf("Number of points: %i \r\n",nsamp);

/* fill tables at every fdacstep fractions of a radian */
/* store a whole cycle of info */
for(fj=0.0;fj<=6.28;fj=fj+fdacstep){
sinev=sin(fj);/* standard SIN function */
sinev= voffset + vpeak * sinev; /* add offset and scale  */
ix=( 65535.0 / span)*(sinev-vlow); /*convert to 0x000-0xFFFF*/
(unsigned short) isinev=  ix;   /* typecast to 16 bits */
wave->ram[ncount]=isinev;  /* download to 4145 waveform buff*/
ncount++;
}
return;
}
```

GE Fanuc Intelligent Platforms Information Centres

Americas:
1 800 322 3616 or 1 256 880 0444

Asia Pacific:
86 10 6561 1561

Europe, Middle East and Africa:
+49 821 5034-0

Additional Resources

For more information, please visit
the GE Fanuc Intelligent Platforms web site at:

www.gefanuc.com

500-024145-000 REV. A.