Model V535

4/8-Axis Stepper Motor Indexer

User's Manual

November 11, 2003

TABLE OF CONTENTS

FIGURES

TABLES

V535 8-axis Stepper Motor Indexer

Provides control for up to 8 stepper motor axes

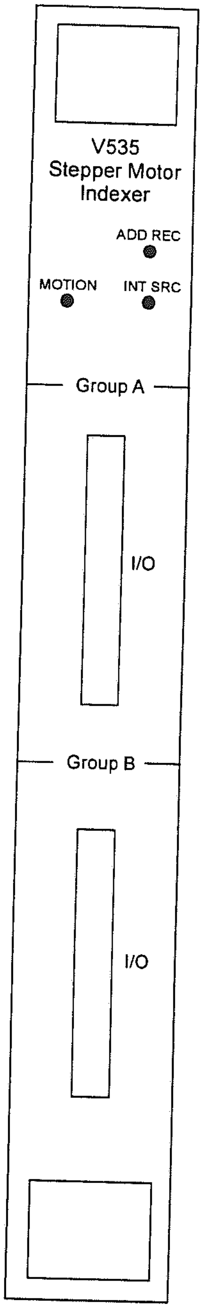
V535

Features

- Single-width, C size, VXI module
- Indexer function provided for 4 or 8 stepper motor axes
- Crystal-controlled programmable pulse rates to 1,550,000 pulses per second
- Programmable linear acceleration and deceleration rates
- All timing internally controlled and not dependent on host computer
- Hard abort activated by limit switches
- Hard and soft abort activated from software
- Auxiliary digital input and output channel provided for each motor axis
- All axes capable of independent setup parameters (velocity, acceleration, etc.) and execution of the motion profiles
- Multi-axes operation supported

Typical Applications

- Control of industrial processes
- Test cell stepper motor control
- Nuclear accelerator control
- General-purpose control of stepper motors



General Description

The V535 is a single-width, C-size, register-based VXIbus module that provides all of the necessary circuitry for the control of 4 or 8 stepper motor axes. The module provides linear or s-curve acceleration and deceleration profiles along with the precise crystal control of the stepping rates. This indexer is designed to be used with one or more external translators that provide the driving current for the stepper motors. The V535 communicates with the translator(s) by logic-level signals. Stepping rates to 1,550,000 pulses/s are supported. All axes can be independently controlled or clustered as "combination motors."

The register-based VXI interface was chosen because an ASCII message-based interface can reduce system throughput dramatically. Ease of use will be attained by the associated VXIplug&play instrument driver. This software driver will provide an ASCII-command-based interface as well as a set of register-based commands to the user application.

A set of 4 or 8 auxiliary digital inputs that can be read by software are provided. A set of 4 or 8 auxiliary digital outputs are also provided. An output can be asserted under one of the following software-selected conditions: Motion has started, motor has received a pre-selected number of pulses, or motion has stopped (by reaching terminal count or by abort).

A VXI interrupt can be generated by limit switch activation, motion complete, or auxiliary digital input activation. These interrupt sources can be individually masked by software

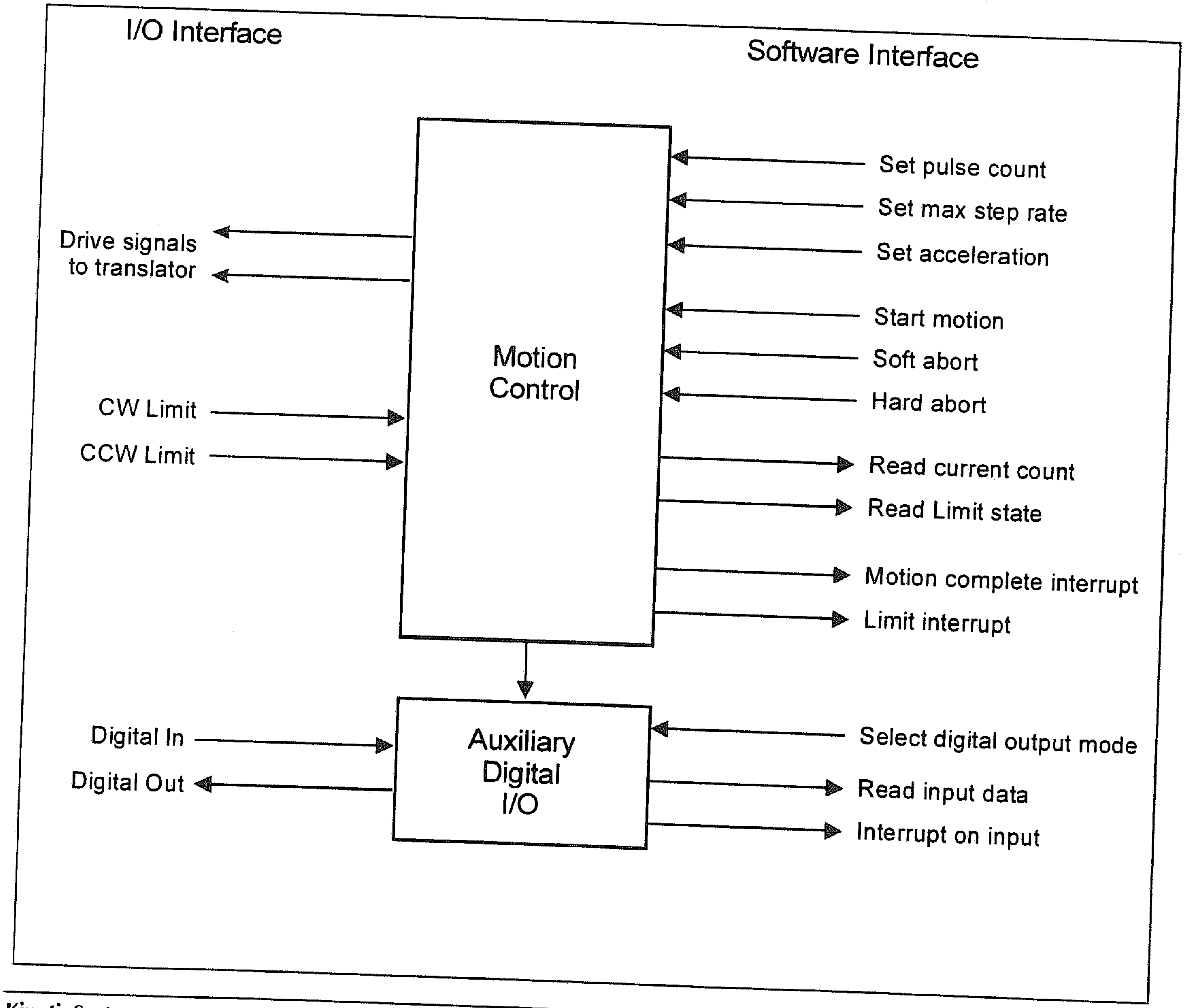
Specifications contained within this data sheet are subject to change without notice.

V535 8-axis Stepper Motor Indexer

Item	Specification
Number of Axes	4 or 8
Type of Control	Stepper motor control via external power translators.
Output Profile Selection	Independent velocity and acceleration profiles per axis
Output Pulse Characteristics	
Signal Level	Open-collector, TTL/CMOS compatible, active low, or RS-422, field selectable
Pulse Width	45% to 55% of any pulse period
Maximum Current Drive	32 mA
Signal Convention	Programmable options (2 output signals per axis): Pulse Output/Direction, *or* CW Pulse Output/CCW Pulse Output
Velocity	
Range	0 to 1,550,000 steps/s, programmable
Resolution	0.0466 steps/s over the entire range, programmable
Starting Velocity Range (linear ramp)	0 to 1,550,000 steps/s, programmable (terminal velocity same as starting velocity)
Starting Velocity Resolution	0.0466 steps/s over the entire range, programmable
Acceleration/deceleration	
Range	0 to 50,000,000 steps/s^2, programmable
Resolution	0.0466 steps/s^2 over the entire range, programmable
Limit Inputs	
Number	Two per axis (CW limit *and* CCW limit)
Signal Level	10 mA isolated current loop
Signal Convention	LOW or HIGH = limit, programmable
Input Delay	2 to 32 ms digital delay, programmable
Abort Characteristics	Hard abort (immediate stop) on *Limit* indication. An interrupt will be generated, if enabled, and the pulse count at abort can be read. The state of the limit inputs also can be read.
Multi-axis Operation	All axes must be able to move at the same time. Multiple axes can be assigned to "combination motors." All axes in a combination motor will be aborted when a limit is activated on one axis.
Software Abort	A "soft abort," executed from software, will start the deceleration phase of an axis or of a combination motor. A "hard abort" from software produces an immediate stop. The ending axis pulse count(s) can then be read the program.
Motion Complete	This action will cause an interrupt, if enabled.
Dynamic Position Polling	The position of any axis can be read by the control program while motion is in progress.
Auxiliary Digital Inputs	
Number	1 per axis (4 or 8 per module)
Signal Level	10 mA isolated current loop
Signal Convention	Current flowing = External stimulus active
Isolation	Each input is opto-isolated with a common isolated return
Action	An interrupt can be generated, and the auxiliary digital input pattern can be read.

Auxiliary Digital Outputs	
Number	1 per axis (4 or 8 per module)
Signal Type	Isolated MOSFET relay
Open-circuit voltage	200 V, line-to-line; 500 V, line-to-ground (maximum)
Current	100 mA (maximum)
Signal Convention	Closed = Internal stimulus active
Isolation	Each output is independently opto-isolated
Action	An output line can be asserted under one of the following software-selected conditions: Motion has started, motor has received a pre-selected number of pulses, *or* motion has stopped (by reaching terminal count or by abort).
I/O Connector Types	One 50-contact high-density connector per 4 axes

Block Diagram



Ordering Information

Model V535-LA11	4-Axis Stepper Motor Indexer
Model V535-LA21	8-Axis Stepper Motor Indexer

061002

About This Manual

Organization

Chapter 1, ***Introduction***, gives you a brief overview of the Model V535, lists items you need to get started, and explains how to safely unpack your module.

Chapter 2, ***Installation and Configuration***, explains how to configure the V535 and correctly insert it into a C-size VXIbus mainframe.

Chapter 3, ***Understanding the V535***, describes the performance of the V535.

Chapter 4, ***Configuration and Operational Registers***, explain how to access and control the V535.

Chapter 5, ***Programming Information***, gives you example setup procedures for preparing the V535 to acquire frequency counts.

The ***Appendices*** provide additional information that may be helpful in learning more about KineticSystems and its products, and in quickly reaching us.

Glossary

Following is a glossary of some of the terms and conventions used throughout this manual:

***** An indicator that a register bit contains low-true data. For example, writing a **"0"** to a bit labeled Enable* would cause a function to be enabled.

A16 Space The first 64 Kbytes of address space, accessible with 16-bit addressing. The configuration registers of VXI devices occupy 64-byte blocks of this address space. The Logical Address of a device determines which 64-block block is associated with that device.

A32 Space The 4 Gbyte address space, accessible with 32-bit addressing. A module can request a block of this address space via information contained in its *Configuration* registers. *Operational* registers, if present, reside in this space.

Configuration Registers Setup registers located in **A16** space. Some are mandatory; some are optional.

D16 A single 16-bit data transfer.

D16 A block transfer of 16-bit words.

D32 A single 32-bit data transfer. Not all Slot-0 controllers support **D32**.

D32 A block transfer of 32-bit words. Not all Slot-0 controllers support **D32** BLK.

Device
One of 255 devices that a VXIbus system can support. The term is often used interchangeably with "module." The distinction is that a VXIbus module can consist of more than one device.

Dynamic Addressing
The VXIbus addressing mode in which the address of a device is stored in a write-able register. See also Static Addressing.

hexadecimal
A base-16 number. The suffix, "h," indicates that a number is hexadecimal. For example, 1Ah = 26_{10}; FFh = 255_{10}; 1000h = 4096_{10}.

Logical Address
A VXIbus module's unique address. A VXIbus system has 254 logical addresss that is available. **"0"** is the address of the Slot-0 controller. **"255"** specifies that dynamic addressing be used to address that module.

Operational Registers
Setup and data-transfer registers that are located in A24 address space.

Resource Manager
Software that sets logical addresss and optimally configures Operational register addresses and memory-block addresses in a system. The manufacturer of the Slot-0 controller provides this software, often referred to as "RESMAN."

Static Addressing
The VXIbus addressing mode in which the address of a device is stored in a switch register. See also Dynamic Addressing.

Chapter 1: Introduction

About the V535

Features

- Indexer function provided for up to 8 stepper motor axes
- Crystal controller programmable pulse rates to 1,550,000 pulses-per-second
- Programmable linear acceleration and deceleration rates
- All timing internally controlled and not dependent on host computer
- Hard abort activated by limit switches
- All axes capable of independent setup parameters (velocity, acceleration, etc.) and execution of the motion profiles
- Multi-axes operation supported through software

Applications

- Control of industrial processes
- Nuclear accelerator control
- General-purpose control of stepper motors

General Description

The V535 is a single-width, C-size, register based VXIbus module that provides all of the necessary circuitry for the control of 4 or 8 stepper motor axes. Options are available with shaft encoder feedback. The module provides linear or S-curve acceleration and deceleration profiles along with the precise crystal control of the stepper rates. This indexer is designed to be used with one or more external translators that provide the driving current for the stepper motors. The V535 communicates with the translator(s) by logic level signals. Stepping rates to 1,550,000 pulses/second are supported. All axes can be independently controlled or as combination motors through software.

The V535 supports both static and dynamic configuration. Access to the data is through memory locations indicated by the Offset Register within the VXIbus Configuration Register set, using A24/A16, D32/D16 data transfers.

Ordering Information

Model V535-LA11	4-Axis Stepper Motor Indexer
Model V535-LB11	4-Axis Stepper Motor Indexer with Encoder
Model V535-LA21	8-Axis Stepper Motor Indexer
Model V535-LB21	8-Axis Stepper Motor Indexer with Encoder

Related products

Getting Started

To set up and use your V535 VXIbus module, you will need most or all of the following:

- The V535 Stepper Motor Indexer module and this User Manual
- Your VXIbus system with its Resource Manager and high-level test and/or application software

Unpacking the V535

The V535 comes in an anti-static bag to avoid electrostatic damage. Electrostatic discharge to the module can damage components on it. Please take the following precautions when unpacking the module:

- Ground yourself with a grounding strap or by touching a grounded object.
- Touch the anti-static package to a metal part of your VXIbus chassis before removing the module from the package.
- Remove the module from the package and inspect the module for damage.
- Do not install the module into the VXIbus chassis until you are satisfied that the module exhibits no obvious mechanical damage and is configured to conform to the desiring operating environment. The next chapter describes installation and configuration.

Chapter 2: Installation and Configuration

Setting the Logical Address Switches

A VXI system can have as many as 255 devices, with each having a unique number in the range from 0 to 254. Eight bits represent the number, which is the Logical Address of the device.

VXIbus defines two concepts of addressing: "static" and "dynamic." All VXIbus devices *must* allow static addressing, in which the address is determined by the setting of a switch register. VXIbus devices may, but are not required to, support dynamic addressing. In dynamic addressing, the Logical Address is stored in a software-addressable register. For reasons discussed in Chapter 4, all KineticSystems VXIbus devices support dynamic as well as static addressing.

Before installing the V535 in the VXIbus chassis, you must set the switch register to an appropriate value. If you wish to employ static addressing you must make sure you set the switch register to a unique value other than 0 or 255. It is a good idea to note module addresses in an accessible log, because if you replace a module, it is very important that the new module have the same address as the replaced one.

If your system employs dynamic addressing, which delegates the task of assigning device addresses to the Resource Manager software, then make sure the address switch is set to 255 (all "**1**"s).

Note: To set a Logical Address bit to "1" depress the bottom segment of the switch.

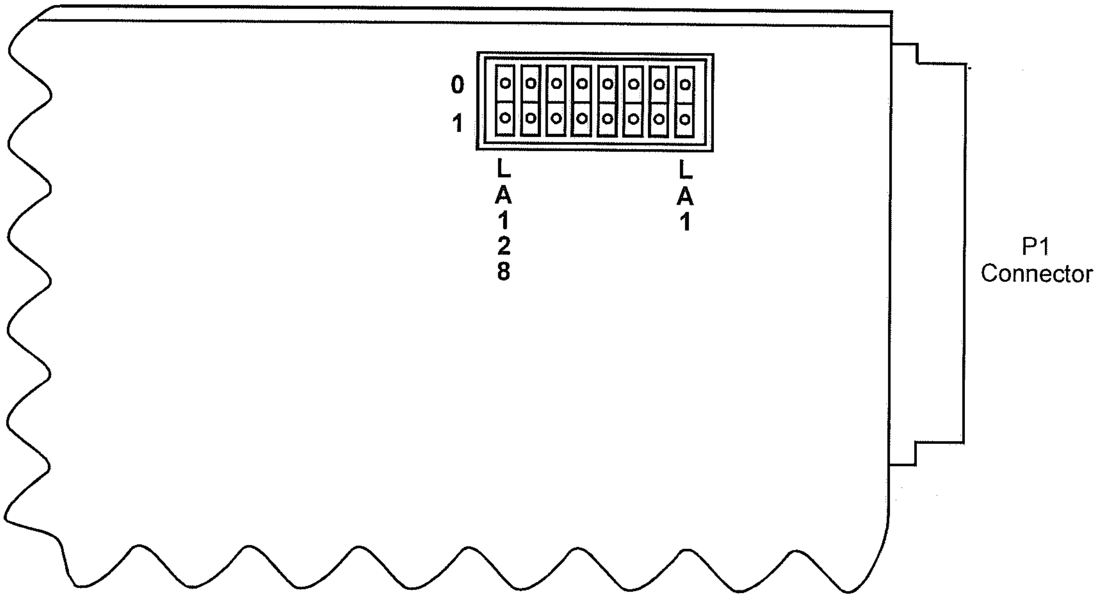
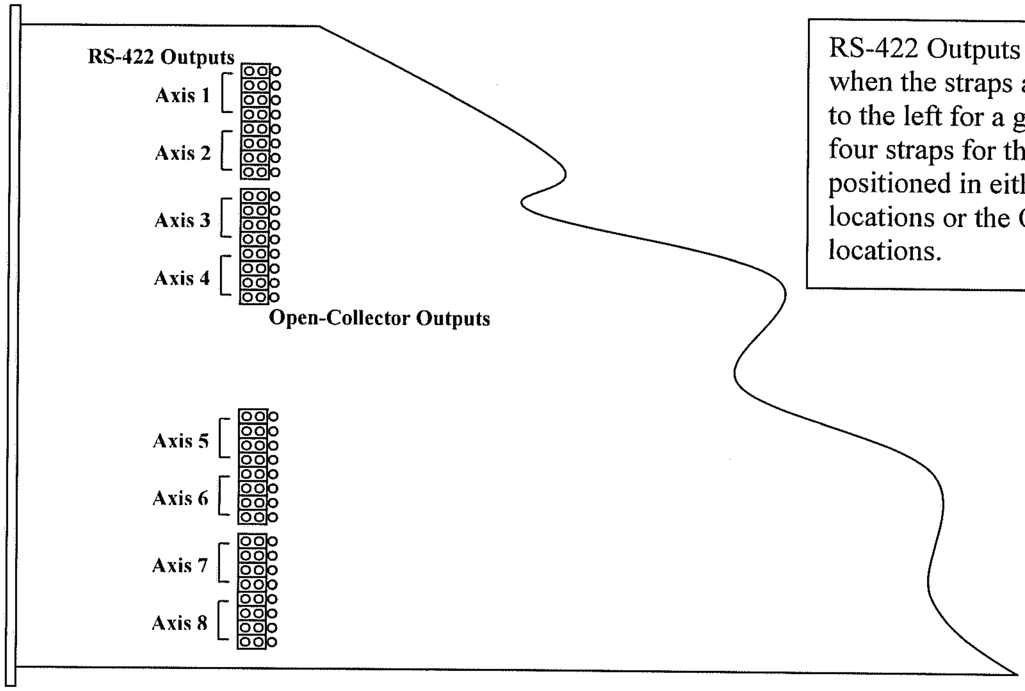


Figure 2-1: V535 Logical Address Switch Locations

Selecting the Stepper Output Signals

The digital outputs from the V535 used to control the stepper motor can be configured to supply either open-collector type outputs or RS-422 outputs. The V535 is configured at the factory to supply the default RS-422 outputs. Strap selections on the module allow the user to reconfigure these outputs from RS-422 to open-collector.

To select the output signal convention, one must remove the ground shield covers on the V535. Once the ground shields are removed, the user then has access to the strap locations for configuring the outputs.



RS-422 Outputs are selected when the straps are positioned to the left for a given axis. All four straps for the axis must be positioned in either the RS-422 locations or the Open-collector locations.

Figure 2-2: V535 Strap Locations

Module Insertion

Before inserting your VXIbus module into the chassis, make sure that the chassis is plugged into electrical power but *not turned on*. The power cord provides a ground connection for the mainframe and protects the equipment and you from electrical harm.

In a VXI system, the Bus Grant and IACK signals are received and transmitted by each of the modules. These signals must be jumpered around any vacant slots in the mainframe. Most current mainframes, including our V194 and V195, contain jumperless backplanes, where the Bus Grant and IACK signals are automatically jumpered when a slot is empty.

If your mainframe does not contain a jumperless backplane, you must position certain jumpers correctly on the chassis backplane to assure that the V535 acknowledges interrupts properly. Remove the Interrupt Acknowledge (IACK) jumper from the slot selected for the V535 and install daisy-chain jumpers in any empty slots between the V535 and the Slot 0 Controller.

You can now insert the V535 into the chassis. Slowly push it in until its plug connectors are resting against the backplane connectors. Then, using evenly distributed pressure, press the module straight in until it seats in the slot and the module front panel is even with the chassis front panel. Tighten the top and bottom screws. *You may now safely apply power to the V535.*

Module Configuration

Setting the VXI Logical Address

You, or your software, must perform two types of module configuration. The first has to do with VXIbus-related items and involves communication with V535 *configuration* registers. The second deals with setting parameters related to module operation and involves communication with V535 *operational* registers.

VXIbus-related configuration includes setting the logical address, specifying the amount of memory space required, specifying where in memory the V535 registers and memory blocks are located, and setting interrupt levels.

VXIbus devices occupy system memory space. The configuration registers for each VXIbus device have 64 bytes of memory space in the upper 16 Kbytes of the 64-kbyte **A16** memory space. Whether you set the 8-bit Logical Address statically in the switch register or dynamically in the Logical Address register, those eight bits determines the base address of the 64-byte block of memory as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	**Logical Address**								**Offset**					

Each 64-byte block contains several registers that supply information about the module, such as the manufacturer, the module identifier (i.e., "207h"), its class (register-based or message-based), serial number, and the amount of memory space it requires.

In addition to **A16** addressing, a VXIbus device can also support **A24** or **A32** addressing. The V535 supports both A16 and A24 addressing.

The operational registers are all in **A24** space. To access them, one must first write a proper offset value to the Offset register in **A16** space.

Refer to Chapter 4 for details relating to the *configuration* and *operational* registers.

Chapter 3: Understanding the V535

Overview

The V535 contains the circuitry necessary to control up to eight stepper motor indexers. The stepper motor outputs of the module are either RS-422 or open-collector for connection to an external stepper motor translator. The external translator is responsible for receiving the digital outputs from the V535 and conditioning these signals for use by the stepper motors.

This module incorporates a 3-chip set to provide a bulk of the control circuitry for the stepper motors. The V535 can be ordered as either a 4-axis module or an 8-axis module. Each 3-chip set can control up to 4 axis. For the 8-axis module, two 3 chip sets are incorporated. Each group of 4-axis is individually controller through the VXI interface.

There are several digital inputs and outputs associated with each axis. Each axis has 2 limit switch inputs, an auxiliary digital input and an auxiliary digital output. The following diagram and discussion explain the functions contained in the V535.

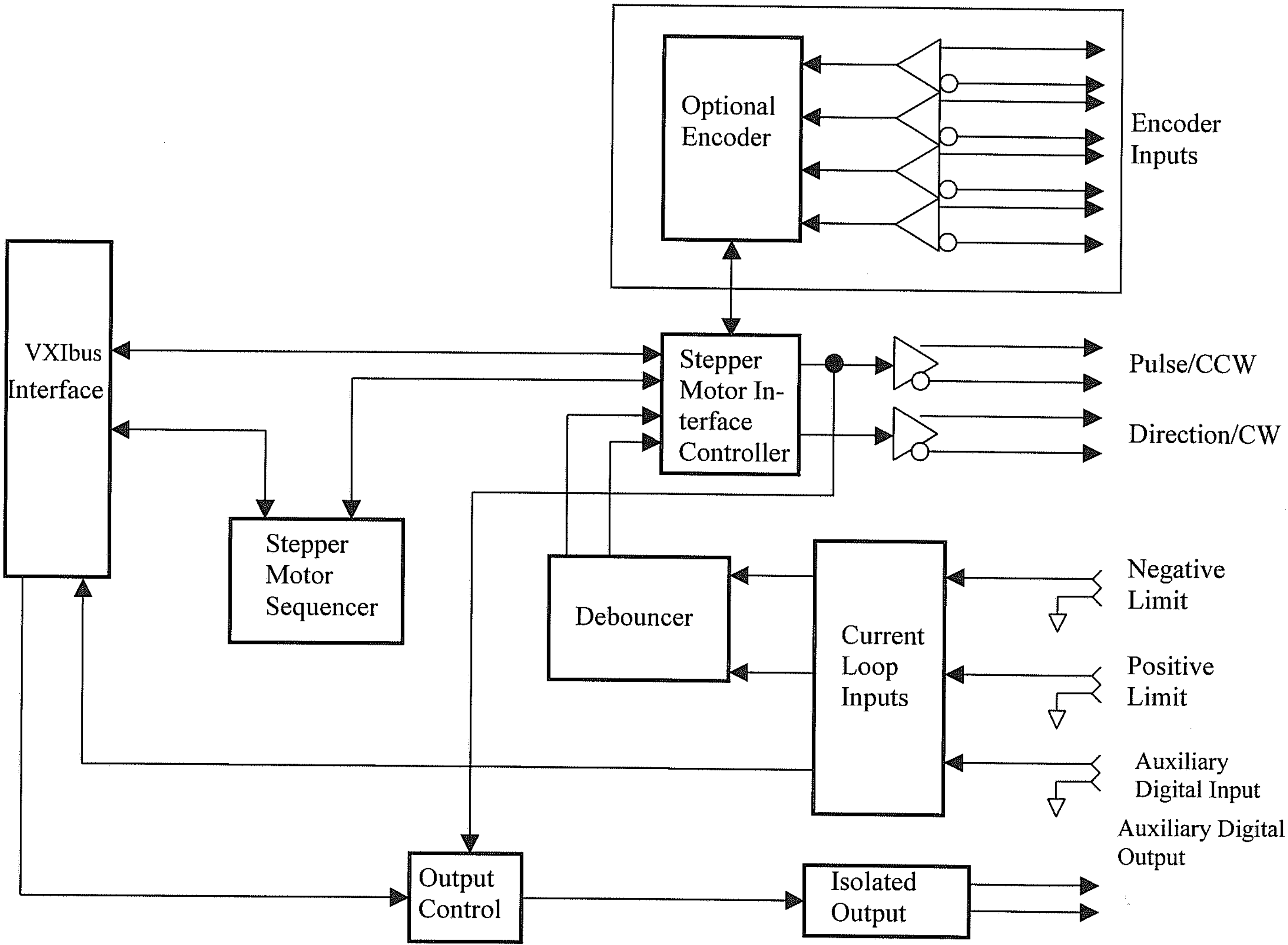


Figure 3-1: V535 Simplified Block Diagram Showing One Channel

Basic Circuit Operation

The previous figure, Figure 3-1, shows a simplified block diagram for one axis on the V535. Each axis consists of the following components.

1.) VXI Interface

2.) Stepper Motor Sequencer

3.) Stepper Motor Interface Controller (SMIC)

4.) Current Loop Inputs

5.) Isolated Outputs

The primary interface to controlling the V535 consists of the VXI Interface, the Stepper Motor Sequencer and the Stepper Motor Interface Controller (SMIC). The VXI Interface block contains all the circuitry necessary to configure, control and monitor the V535. The registers for control of the V535 are mapped into A24 space in VXI and are accessed by either D16 and/or D32 data transfers. The VXI interface communicates with both the Stepper Motor Sequencer and the Stepper Motor Interface Controller. The Stepper Motor Interface Controller (SMIC) consists of a 2-chip set manufactured by Performance Motion Devices. Their part number for the two-chip set is MC1451A. Optionally, an encoder may be ordered for the V535 (See ordering information) which is a third chip for the chip set. These three devices make up the primary interface to stepper motors.

The Stepper Motor Interface Controller is responsible for executing all commanded operations. The chip-set controls all aspects of the stepper motor operation including acceleration, velocity, profiling and contouring. The outputs of this chipset are buffered for direct connection to the stepper motor translator for conditioning before application to the motors. The output signals from the V535 are factory configured to be RS-422 outputs, but can be reconfigured by strap options on the V535 for open-collector. For open-collector outputs, the resistor terminations must be supplied external to the V535. The output signals can be programmed to operate as either a DIRECTION and PULSE pair or a CLOCKWISE and COUNTER-CLOCKWISE pair. The optional encoder on the V535 requires RS-422 balanced inputs.

The Stepper Motor Interface Controller can be communicated to directly with the VXI interface, or indirectly through the Stepper Motor Sequencer. The Stepper Motor Sequencer can be programmed to work in one of two modes. The first mode is virtually transparent to the Stepper Motor Interface Controller. All commands given to the V535 are directly transferred to the Stepper Motor Interface Controller. This mode places the burden on the host computer to monitor each transfer of a data word to/from the Stepper Motor Interface Controller. This mode can be useful when debugging application software that makes use of the Stepper Motor Sequencer.

The second mode for communicating with the Stepper Motor Interface Controller makes use of the Stepper Motor Sequencer. The Stepper Motor Sequencer can alleviate the host computer from participating in the handshaking (polling) protocol to/from the Stepper Motor Interface Controller. A macro style command can be directed at a particular axis through the Stepper Motor Sequencer. This macro command instructs the sequencer on the V535 to perform the indicated operation. This operation can be a write operation, a read operation, or a control operation. Write operations are those operations in which a command is executed to the Stepper Motor Interface Controller followed by its required data necessary to execute the operation. A read operation is a command that returns data following the command phase of the operation. A control operation requires no data transfer and the only data transferred to the Stepper Motor Interface Controller is the command.

The data path to/from the Stepper Motor Interface Controller is only 8 bits in width. Command data to the Stepper Motor Controller is 8 bits in width, therefore requiring one write operation to the chipset.

After the command phase, data may be transferred to/from the chipset. This data can consist of either one or two 16-bit words (one 32-bit word). Since the maximum data word size is 32-bits, up to 4 byte cycles must occur to the Stepper Motor Interface Controller. Between each phase of data transfer and between each byte of data transferred, the Stepper Motor Interface Controller chipset must be polled to see if it is ready to accept/provide additional data. The Stepper Motor Sequencer facilitates all data transactions to the Stepper Motor Interface Controller by performing these tasks automatically. For a write operation, the user specifies the command byte, the write data, number of data words, and the direction of transfer. The Stepper Motor Sequencer then executes all data transfers, polling for chipset ready, and returns a flag when the operation is complete. For a read operation, the user specifies the command byte, number of returned data words, and the direction of transfer. Once specified, the Stepper Motor Sequencer executes the commands and returns a flag when the operation is complete. When finished, the application software can retrieve the data sent from the chipset. Optionally, the Stepper Motor Sequencer can be instructed to return the checksum for the command. After an operation is executed, the Stepper Motor Sequencer will retrieve the latest checksum from the chipset and store the result in a register accessible by VXI. The user can then verify the integrity of the requested operation.

Another use of the Stepper Motor Sequencer is for executing the same command sequence to multiple axes. These instructions are executed by using the Multi-Axis Command Register and can be executed for non-data transfer instructions (commands only). This provides a quick mechanism for executing the same control operation to specified axes without host computer intervention. A mask pattern is used to specify on which axis the operation is to be executed. A similar command is available at the chipset command level itself, but does not span all 8 axes, it only acts on the 4 axes controlled by the chipset.

There are three current loop inputs for each axis in theV535. These inputs are in the form of isolated 10-milliampere current loops. Two of the inputs are run through a digital debouncer to remove false detection due to noisy environments. The debouncer is programmable from 2 milliseconds to 32 milliseconds. These two inputs are then fed into the Stepper Motor Controller chipset to provide the Clockwise and Counterclockwise limit inputs. The third current loop input on an axis is general purpose in nature and is defined by the user. This input is not debounced and can be read directly through the VXI interface. Optionally, this input can be configured to generate a VXI interrupt when asserted. All three of these inputs, on a per axis basis can be programmed for either polarity.

There is also one digital output per axis that can be used to control external V535 operation. This output is in the form of an isolated photovoltaic (isolated MOSFET) relay. This relay can be activated by one of four sources. These sources include by programmed I/O, by the axis motion starting, by the axis motion ceasing, or by a pulse counter. The pulse counter is a 24-bit counter that is preloaded with the number of pulses that the V535 should output before asserting the digital output. This can be useful for synchronizing external events with the stepper motor pulse train.

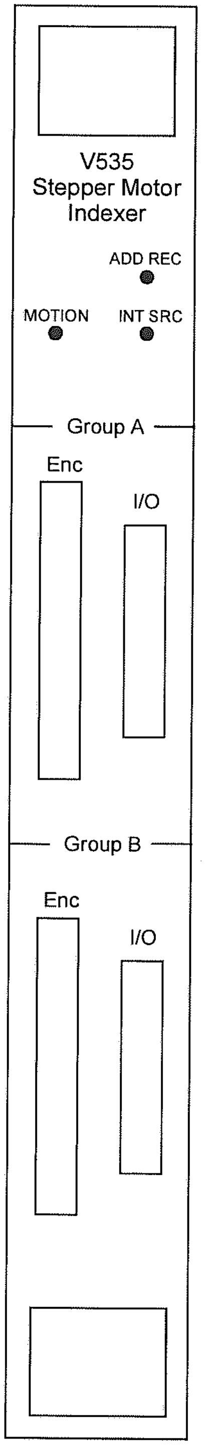
V535 Specifications

Item	Specification
Number of Axes	4 or 8
Type of Control	Stepper motor control via external power translators. Options with encoder feedback are available.
Output Profile Selection	Independent velocity and acceleration profiles per axis
Output Pulse Characteristics	
Signal Level	Open-collector, TTL/CMOS compatible, active low, or RS-422, field selectable
Pulse Width	45% to 55% of any pulse period
Maximum Current Drive	32 mA
Signal Convention	Programmable options (2 output signals per axis): Pulse Output/Direction, *or* CW Pulse Output/CCW Pulse Output
Encoder Inputs (Encoder option)	
Signal Level	RS-422 with 100 Ohm termination in the V535
Minimum Pulse Width	480 ns (signal TRUE or FALSE)
Velocity	
Range	0 to 1,550,000 steps/s, programmable
Resolution	0.0466 steps/s over the entire range, programmable
Starting Velocity Range (linear ramp)	0 to 1,550,000 steps/s, programmable (terminal velocity same as starting velocity)
Starting Velocity Resolution	0.0466 steps/s over the entire range, programmable
Acceleration/deceleration	
Range	0 to 50,000,000 steps/s^2, programmable
Resolution	0.0466 steps/s^2 over the entire range, programmable
Limit Inputs	
Number	Two per axis (CW limit *and* CCW limit)
Signal Level	10 mA isolated current loop
Signal Convention	LOW or HIGH = limit, programmable
Input Delay	2 to 32 ms digital delay, programmable
Abort Characteristics	Hard abort (immediate stop) on *Limit* indication. An interrupt will be generated, if enabled, and the pulse count at abort can be read. The state of the limit inputs also can be read.
Software Abort	A "soft abort," executed from software, will start the deceleration phase of an axis or of a combination motor. A "hard abort" from software produces an immediate stop. The ending axis pulse count(s) can then be read the program.
Motion Complete	This action will cause an interrupt, if enabled.
Dynamic Position Polling	The position of any axis can be read by the control program while motion is in progress.
Auxiliary Digital Inputs	
Number	1 per axis (4 or 8 per module)
Signal Level	10 mA isolated current loop
Signal Convention	Current flowing = External stimulus active
Isolation	Each input is opto-isolated with a common isolated return
Action	An interrupt can be generated, and the auxiliary digital input pattern can be read.

Item	Specification
Auxiliary Digital Outputs	
Number	1 per axis (4 or 8 per module)
Signal Type	Isolated MOSFET relay
Open-circuit voltage	200 V, line-to-line; 500 V, line-to-ground (maximum)
Current	100 mA (maximum)
Signal Convention	Closed = Internal stimulus active
Isolation	Each output is independently opto-isolated
Action	An output line can be asserted under one of the following software-selected conditions: Motion has started, motor has received a pre-selected number of pulses, *or* motion has stopped (by reaching terminal count or by abort).

Table 3-1: Specifications

Front Panel



LEDs

Motion

Illuminated when the module is causing at least one of the stepper motor axes to be in motion.

Add Rec

Illuminated when the module is being accessed.

Int Src

Illuminated as long as the V535 has an interrupt pending.

Connectors

Group A I/O

This 50-contact "D" connector with pins provides the connections to the 4 axes associated with Group A. The signals on this connector include pulse/direction, limit switch, and auxiliary digital I/O.

Group B I/O

This 50-contact "D" connector with pins provides the connections to the 4 axes associated with Group B. The signals on this connector include pulse/direction, limit switch, and auxiliary digital I/O.

Group A Encoder

This 36-contact "D" connector with pins provides the connections to the 4 axes associated with Group A. The signals on this connector are used for connection to the encoder inputs.

Group B Encoder

This 36-contact "D" connector with pins provides the connections to the 4 axes associated with Group B. The signals on this connector are used for connection to the encoder inputs.

Group A I/O and Encoder Connector Pinouts

Axis 3 CW/Pulse Return	26	1	Axis 1 CW/Pulse Signal
Axis 3 CCW/Direction Signal	27	2	Axis 1 CW/Pulse Return
Axis 3 CCW/Direction Return	28	3	Axis 1 CCW/Direction Signal
Axis 3 CW Limit Signal	29	4	Axis 1 CCW/Direction Return
Axis 3 CW Limit Return	30	5	Axis 1 CW Limit Signal
Axis 3 CCW Limit Signal	31	6	Axis 1 CW Limit Return
Axis 3 CCW Limit Return	32	7	Axis 1 CCW Limit Signal
Axis 3 Digital Input Signal	33	8	Axis 1 CCW Limit Return
Axis 3 Digital Input Return	34	9	Axis 1 Digital Input Signal
Axis 3 Digital Output Signal	35	10	Axis 1 Digital Input Return
Axis 3 Digital Output Return	36	11	Axis 1 Digital Output Signal
Axis 4 CW/Pulse Signal	37	12	Axis 1 Digital Output Return
Axis 4 CW/Pulse Return	38	13	Axis 2 CW/Pulse Signal
Axis 4 CCW/Direction Signal	39	14	Axis 2 CW/Pulse Return
Axis 4 CCW/Direction Return	40	15	Axis 2 CCW/Direction Signal
Axis 4 CW Limit Signal	41	16	Axis 2 CCW/Direction Return
Axis 4 CW Limit Return	42	17	Axis 2 CW Limit Signal
Axis 4 CCW Limit Signal	43	18	Axis 2 CW Limit Return
Axis 4 CCW Limit Return	44	19	Axis 2 CCW Limit Signal
Axis 4 Digital Input Signal	45	20	Axis 2 CCW Limit Return
Axis 4 Digital Input Return	46	21	Axis 2 Digital Input Signal
Axis 4 Digital Output Signal	47	22	Axis 2 Digital Input Return
Axis 4 Digital Output Return	48	23	Axis 2 Digital Output Signal
Ground	49	24	Axis 2 Digital Output Return
Ground	50	25	Axis 3 CW/Pulse Signal

Axis 4 Quad A Signal	19	1	Axis 1 Quad A Signal
Axis 4 Quad A Return	20	2	Axis 1 Quad A Return
Axis 4 Quad B Signal	21	3	Axis 1 Quad B Signal
Axis 4 Quad B Return	22	4	Axis 1 Quad B Return
Ground	23	5	Ground
+5 Volts	24	6	+5 Volts
Reserved	25	7	Axis 2 Quad A Signal
Reserved	26	8	Axis 2 Quad A Return
Reserved	27	9	Axis 2 Quad B Signal
Reserved	28	10	Axis 2 Quad B Return
Reserved	29	11	Ground
Reserved	30	12	+5 Volts
Reserved	31	13	Axis 3 Quad A Signal
Reserved	32	14	Axis 3 Quad A Return
Reserved	33	15	Axis 3 Quad B Signal
Reserved	34	16	Axis 3 Quad B Return
Reserved	35	17	Ground
Reserved	36	18	+5 Volts

Group B I/O and Encoder Connector Pinouts

Axis 7 CW/Pulse Return	26	1	Axis 5 CW/Pulse Signal
Axis 7 CCW/Direction Signal	27	2	Axis 5 CW/Pulse Return
Axis 7 CCW/Direction Return	28	3	Axis 5 CCW/Direction Signal
Axis 7 CW Limit Signal	29	4	Axis 5 CCW/Direction Return
Axis 7 CW Limit Return	30	5	Axis 5 CW Limit Signal
Axis 7 CCW Limit Signal	31	6	Axis 5 CW Limit Return
Axis 7 CCW Limit Return	32	7	Axis 5 CCW Limit Signal
Axis 7 Digital Input Signal	33	8	Axis 5 CCW Limit Return
Axis 7 Digital Input Return	34	9	Axis 5 Digital Input Signal
Axis 7 Digital Output Signal	35	10	Axis 5 Digital Input Return
Axis 7 Digital Output Return	36	11	Axis 5 Digital Output Signal
Axis 8 CW/Pulse Signal	37	12	Axis 5 Digital Output Return
Axis 8 CW/Pulse Return	38	13	Axis 6 CW/Pulse Signal
Axis 8 CCW/Direction Signal	39	14	Axis 6 CW/Pulse Return
Axis 8 CCW/Direction Return	40	15	Axis 6 CCW/Direction Signal
Axis 8 CW Limit Signal	41	16	Axis 6 CCW/Direction Return
Axis 8 CW Limit Return	42	17	Axis 6 CW Limit Signal
Axis 8 CCW Limit Signal	43	18	Axis 6 CW Limit Return
Axis 8 CCW Limit Return	44	19	Axis 6 CCW Limit Signal
Axis 8 Digital Input Signal	45	20	Axis 6 CCW Limit Return
Axis 8 Digital Input Return	46	21	Axis 6 Digital Input Signal
Axis 8 Digital Output Signal	47	22	Axis 6 Digital Input Return
Axis 8 Digital Output Return	48	23	Axis 6 Digital Output Signal
Ground	49	24	Axis 6 Digital Output Return
Ground	50	25	Axis 7 CW/Pulse Signal

Axis 8 Quad A Signal	19	1	Axis 5 Quad A Signal
Axis 8 Quad A Return	20	2	Axis 5 Quad A Return
Axis 8 Quad B Signal	21	3	Axis 5 Quad B Signal
Axis 8 Quad B Return	22	4	Axis 5 Quad B Return
Ground	23	5	Ground
+5 Volts	24	6	+5 Volts
Reserved	25	7	Axis 6 Quad A Signal
Reserved	26	8	Axis 6 Quad A Return
Reserved	27	9	Axis 6 Quad B Signal
Reserved	28	10	Axis 6 Quad B Return
Reserved	29	11	Ground
Reserved	30	12	+5 Volts
Reserved	31	13	Axis 7 Quad A Signal
Reserved	32	14	Axis 7 Quad A Return
Reserved	33	15	Axis 7 Quad B Signal
Reserved	34	16	Axis 7 Quad B Return
Reserved	35	17	Ground
Reserved	36	18	+5 Volts

Front-panel Connector Pinout

The following chart shows the connector pinout for stepper motor indexer control signals on Axis 1 through 4 on the V535. This connector is labeled Group A I/O on the front panel.

Pin Number	Axis	Signal	Axis	Pin Number	Signal
1	1	CW / Pulse Signal *	1	2	CW / Pulse Return *
3	1	CCW / Direction Signal *	1	4	CCW / Direction Return *
5	1	CW Limit Signal	1	6	CW Limit Return (IsoGround)
7	1	CCW Limit Signal	1	8	CCW Limit Return (IsoGround)
9	1	Digital Input Signal	1	10	Digital Input Return (IsoGround)
11	1	Digital Output Signal	1	12	Digital Output Return
13	2	CW / Pulse Signal *	2	14	CW / Pulse Return *
15	2	CCW / Direction Signal *	2	16	CCW / Direction Return *
17	2	CW Limit Signal	2	18	CW Limit Return (IsoGround)
19	2	CCW Limit Signal	2	20	CCW Limit Return
21	2	Digital Input Signal	2	22	Digital Input Return (IsoGround)
23	2	Digital Output Signal	2	24	Digital Output Return
25	3	CW /Pulse Signal *	3	26	CW / Pulse Return *
27	3	CCW / Pulse Signal *	3	28	CCW / Direction Return *
29	3	CW Limit Signal	3	30	CW Limit Return (IsoGround)
31	3	CCW Limit Signal	3	32	CCW Limit Return (IsoGround)
33	3	Digital Input Signal	3	34	Digital Input Return (IsoGround)
35	3	Digital Output Signal	3	36	Digital Output Return
37	4	CW / Pulse Signal *	4	38	CW / Pulse Return *
39	4	CCW / Direction Signal *	4	40	CCW / Direction Return *
41	4	CW Limit Signal	4	42	CW Limit Return (IsoGround)
43	4	CCW Limit Signal	4	44	CCW Limit Return (IsoGround)
45	4	Digital Input Signal	4	46	Digital Input Return (IsoGround)
47	4	Digital Output Signal	4	48	Digital Output Return
49	-	Ground	-	50	Ground

Table 3-2: Group A I/O Connector Pinout for Stepper Motor Indexer Control Signals on Axis 1-4

The following chart shows the connector pinout for the stepper motor indexer controller signals for Axis 5 through 8 on the V535. This connector is labeled Group B I/O on the front panel.

Pin Number	Axis	Signal	Axis	Pin Number	Signal
1	5	CW / Pulse Signal *	5	2	CW / Pulse Return *
3	5	CCW / Direction Signal *	5	4	CCW / Direction Return *
5	5	CW Limit Signal	5	6	CW Limit Return (IsoGround)
7	5	CCW Limit Signal	5	8	CCW Limit Return (IsoGround)
9	5	Digital Input Signal	5	10	Digital Input Return (IsoGround)
11	5	Digital Output Signal	5	12	Digital Output Return
13	6	CW / Pulse Signal *	6	14	CW / Pulse Return *
15	6	CCW / Direction Signal *	6	16	CCW / Direction Return *
17	6	CW Limit Signal	6	18	CW Limit Return (IsoGround)
19	6	CCW Limit Signal	6	20	CCW Limit Return
21	6	Digital Input Signal	6	22	Digital Input Return (IsoGround)
23	6	Digital Output Signal	6	24	Digital Output Return
25	7	CW /Pulse Signal *	7	26	CW / Pulse Return *
27	7	CCW / Pulse Signal *	7	28	CCW / Direction Return *
29	7	CW Limit Signal	7	30	CW Limit Return (IsoGround)
31	7	CCW Limit Signal	7	32	CCW Limit Return (IsoGround)
33	7	Digital Input Signal	7	34	Digital Input Return (IsoGround)
35	7	Digital Output Signal	7	36	Digital Output Return
37	8	CW / Pulse Signal *	8	38	CW / Pulse Return *
39	8	CCW / Direction Signal *	8	40	CCW / Direction Return *
41	8	CW Limit Signal	8	42	CW Limit Return (IsoGround)
43	8	CCW Limit Signal	8	44	CCW Limit Return (IsoGround)
45	8	Digital Input Signal	8	46	Digital Input Return (IsoGround)
47	8	Digital Output Signal	8	48	Digital Output Return
49	-	Ground	-	50	Ground

Table 3-3: Group B I/O Connector Pinout for Stepper Motor Indexer Controller Signals for Axis 5-8

The following chart shows the connector pinout for the shaft encoder inputs for Axis 1 through 4 on the V535. This connector is labeled Encoder on the front panel.

Pin Number	Axis	Signal	Axis	Pin Number	Signal
1	1	Quad A Signal	1	2	Quad A Return
3	1	Quad B Signal	1	4	Quad B Return
5	-	Ground	-	6	+5 Volts
7	2	Quad A Signal	2	8	Quad A Return
9	2	Quad B Signal	2	10	Quad B Return
11	-	Ground	-	12	+5 Volts
13	3	Quad A Signal	3	14	Quad A Return
15	3	Quad B Signal	3	16	Quad B Return
17	-	Ground	-	18	+5 Volts
19	4	Quad A Signal	4	20	Quad A Return
21	4	Quad B Signal	4	22	Quad B Return
23	-	Ground	-	24	+5 Volts
25	-	Reserved	-	26	Reserved
27	-	Reserved	-	28	Reserved
29	-	Reserved	-	30	Reserved
31	-	Reserved	-	32	Reserved
33	-	Reserved	-	34	Reserved
35	-	Reserved	-	36	Reserved

Table 3-4: Encoder Connector Pinout for Shaft Encoder Inputs for Axis 1-4

The following chart shows the connector pinout for the shaft encoder inputs for Axis 5 through 8 on the V535. This connector is labeled Encoder on the front panel.

Pin Number	Axis	Signal	Axis	Pin Number	Signal
1	5	Quad A Signal	5	2	Quad A Return
3	5	Quad B Signal	5	4	Quad B Return
5	-	Ground	-	6	+5 Volts
7	6	Quad A Signal	6	8	Quad A Return
9	6	Quad B Signal	6	10	Quad B Return
11	-	Ground	-	12	+5 Volts
13	7	Quad A Signal	7	14	Quad A Return
15	7	Quad B Signal	7	16	Quad B Return
17	-	Ground	-	18	+5 Volts
19	8	Quad A Signal	8	20	Quad A Return
21	8	Quad B Signal	8	22	Quad B Return
23	-	Ground	-	24	+5 Volts
25	-	Reserved	-	26	Reserved
27	-	Reserved	-	28	Reserved
29	-	Reserved	-	30	Reserved
31	-	Reserved	-	32	Reserved
33	-	Reserved	-	34	Reserved
35	-	Reserved	-	36	Reserved

Table 3-5: Encoder Connector Pinout for Shaft Encoder Inputs for Axis 5-8

Chapter 4: Configuration and Operational Registers

Address Space

VXIbus uses the VMEbus protocol for data transfer and therefore supports 32-bit addressing to access I/O slave devices. 32-bit addressing provides direct access to memory space of four Gigabytes.

Slave devices such as VXIbus data acquisition modules exist for a variety of purposes and can be simple or very complex. Communication between host and slave can require access to several registers in one device or access to many Mbytes of memory in another. ("Devices" and "modules" are terms often used interchangeably. The distinction is that more than one VXIbus device *can* reside in a VXIbus module. However, there is generally one device per module.)

To minimize the amount of address-decoding hardware needed, simpler slave devices use addressing modes that fully decode only 16 or 24 address lines rather than 32. Therefore, there are three defined addressing modes...**A16**, **A24** and **A32**...having address spaces of 64 Kbytes, 16 Mbytes and 4 Gbytes, respectively.

All VXIbus devices have registers located within 64-byte blocks in **A16** address space and therefore support **A16** addressing. Devices requiring no more than 64 bytes of address space need only support **A16** addressing. Devices needing more than the 64 bytes to accommodate additional registers or blocks of memory *must* also support **A24** or **A32** addressing, but not both.

VXIbus devices that use **A24** or **A32** addressing modes are required to have four registers in **A16** space for parameter definition. One such parameter is Required Memory, which uses four bits (*m*) to specify the size of the memory in **A24** or **A32** space required by the device. A device *may not* use more than one-half of the memory space, and it *should not* use more than one-fourth. Table 4-1 shows the relationship between the four-bit parameter, *m,* and the memory required by the device. Note that *m = 0* defines the case for maximum usage, i.e.; half of the memory space. Required Memory is specified in bits 15 – 12 in the Device Type register at offset 02h.

m	Required Memory				*m*	Required Memory			
	A24		A32			A24		A32	
0	8	Mbytes	2	Gbytes	**8**	32	Kbytes	8	Mbytes
1	4	“	1	“	**9**	16	“	4	“
2	2	“	512	Mbytes	**10**	8	“	2	“
3	1	“	256	“	**11**	4	“	1	“
4	512	Kbytes	128	“	**12**	2	“	512	Kbytes
5	256	“	64	“	**13**	1	“	256	“
6	128	“	32	“	**14**	512	bytes	128	“
7	64	“	16	“	**15**	256	“	64	“

Table 4-1: Relationship between the "*m*" Parameter and Required Memory

One of the four registers is the Offset register, which is needed only for devices using **A24** or **A32** address space. This 16-bit read/write register defines the base address of the device's **A24** or **A32** operational registers. The *m+1* most significant bits of the Offset register provide the values of the *m+1* most significant bits of the device's **A24** or **A32** register addresses, where *m* is as defined in Table 4-1 above.

Static and Dynamic Configuration

A VXIbus system can have up to 255 devices. Therefore, eight bits define the device address, which is called the "Logical Address." The Logical Address can be "static" or "dynamic." A static address resides in an 8-bit switch register; a dynamic address resides in a write-only register. Setting the switch register to 255 (all "**1**"s) causes dynamic addressing to be enabled. Any other setting enables static addressing, in which case the value held in the switch register is the Logical Address.

With the Logical Address set to 255, a device responds to accesses at address 255 only when the MODID line is asserted as a qualifier by the Slot-0 controller. After a new Logical Address is written to the device, the device responds to the new address independent of the state of the MODID line.

For data acquisition and control applications, dynamic configuration is an important concept. A system often contains more than one module of a given type, and it can be easy, and sometimes desirable, to swap positions of two modules after removing them from the mainframe. If dynamic configuration is not employed, one must make sure that the switch register is correctly set when inserting or re-inserting a device. Dynamic configuration greatly simplifies system setup, since the software can assure that the devices are located in the desired slots. Dynamic configuration also allows a system's Resource Manager to configure memory usage optimally in a system.

Communication Protocol

VXIbus allows communication over the backplane by either register-based or message-based protocols. With register-based protocol, the communication is via an 8-, 16- or 32-bit parallel path directly to I/O registers within the modules. With message-based protocol, an ASCII interpreter is included on each module, and the binary representations of ASCII characters are transmitted over the backplane. The advantage of message-based protocol is that English-like commands and responses can be used.

High-performance data acquisition and control modules are usually register-based because the data throughput is usually several orders of magnitude greater than with message-based devices. All Kinetic-Systems VXIbus devices are register-based.

Register Addressing

The user assigns each device in a VXIbus system a unique number between 1 and 254. This 8-bit number, called the Logical Address, defines the base address for the VXIbus device registers located on the module. Each device has a 64-byte block of memory reserved for these registers. The memory blocks, called configuration space, are located in the upper 16 Kbytes of the 64-kbyte **A16** address space.

Every device has at least three configuration registers: ID / Logical Address, Device Type, and Status / Control. Modules using **A24** or **A32** addressing must also have an Offset register. The rest of the 64-byte block can contain registers or memory appropriate for the operation of the specific device.

A device's Logical Address occupies bits 13 - 6 of the register address. Bits 15 and 14 of the address are both "**1**'s," and the base address of the register block is therefore:

V*40h+C000h

where *V* is the Logical Address of the device and C000h is the starting address of the top 16-kbyte block.

The address of a specific register is the base address plus an offset address. The offset is bits 5 - 0 of the register address and ranges from 00h to 3Eh.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	Logical Address								Offset					

The V535 also uses *operational* registers in **A24 space; therefore,** it is an "Extended" register-based device.

Required Configuration Registers

The four required VXIbus registers are ID / Logical Address, Device Type, Status / Control, and Offset. You can access these registers by **D16** transfers only.

ID Register 00h

This read-only register returns 4F29h.

Fields are Device Classification, Addressing Mode and Manufacturer ID.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	1
	Class = Extended		**Addressing Mode = A24**		**KineticSystems' Manufacturer ID = F29h (3881)**											

Device Classification — Bits 15 and 14

00	Memory device
01	**Extended device**
10	Message-based device
11	Register-based device

The V535 is an *Extended device.*

Addressing Mode — Bits 13 and 12

00	**A24**
01	A32
10	Reserved
11	A16

The V535 uses *A24 addressing*.

Manufacturer ID — Bits 11 through 0

KineticSystems' Manufacturer ID is *3881*, which corresponds to F29h.

Logical Address Register 00h

This write-only register holds the Logical Address. In systems using Dynamic Configuration, the system Resource Manager uses this register to set the Logical Address of the device.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write-only	Not Used								Logical Address							

Logical Address — Bits 7 through 0

Device Type Register 02h

This read-only register contains the Required Memory and Model Code for the V535. It returns F535h.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	1	1	1	1	0	1	0	1	0	0	1	1	0	1	0	1
	Required Memory (*m*)				Model Code = 535h											

Required Memory Bits 15 – 12

A value of **Fh** is returned (*m* = 15 decimal), indicating that the V535 is allocated 256 bytes in **A24** space.

Model Code Bits 11 – 0

The model code for the V535 is 535h.

Status Register 04h

This read-only register provides binary information about the status of the V535.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	A24 Active	MODID*	Not Used										Ready	Passed	Sysfail Inhibit	Soft Reset

A24 Active Bit 15

"**1**" in this field indicates that the **A24** registers of the V535 can be accessed. This bit reflects the state of the Control register's **A24** Enable bit.

MODID* Bit 14

"**1**" in this field indicates that the V535 is *not* selected via the P2 MODID line. A "**0**" indicates that the device is selected by a high state on the P2 MODID line. The Resource Manager uses this bit to configure the V535 dynamically.

Ready Bit 3

"**1**" in this field indicates that the registers have been successfully initialized. The V535 is ready for access.

Passed Bit 2

"**1**" in this field indicates that the device self-test has passed. A "**0**" indicates that the V535 has failed—or is currently executing—its self-test.

Sysfail Inhibit Bit 1

"**1**" in this field indicates that the V535 is disabled from driving the SYSFAIL* line. This bit reflects the state of the Sysfail Inhibit line in the Control register.

Soft Reset Bit 0

"**1**" in this field indicates that the V535 is in a reset state. While in this state, the V535 will allow access only to its Configuration registers.

Control Register *04h*

This write-only register causes execution of specific actions by the V535.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write-only	A24 Enable	Not Used													Sysfail Inhibit	Soft Reset

A24 Enable *Bit 15*

Setting this bit to "**1**" enables access of the **A24** registers of the V535.

Sysfail Inhibit *Bit 1*

Setting this bit to "**1**" disables the V535 from driving the SYSFAIL* line.

Soft Reset *Bit 0*

Setting this bit to "**1**" forces the V535 into a reset state.

Offset Register *06h*

This read/write register determines and reports the device base address in **A24** memory space.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/write	Base Address in A24 memory space								0	0	0	0	0	0	0	0

Additional Configuration Registers

Additional configuration registers are:

- Attribute register
- Serial Number High & Low registers
- Version Number register
- Interrupt Status register
- Interrupt Control register
- Subclass register, and
- Suffix High and Low registers.

Note that you can access these registers by **D16** transfers only.

Attribute Register *08h*

This read-only register provides low-true information about the V535's interrupt handling capabilities. A read of this register returns FFFAh.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
	Reserved													Interrupt Capability*	Interrupt Handler Control*	Interrupt Status Reporting*

Reserved *Bits 15 - 3*

These bits are reserved for future use and return "1"s when read.

Interrupt Capability Bit 2*

"**0**" signifies that the V535 is capable of generating interrupts.

Interrupt Handler Control Bit 1*

"**1**" indicates that the V535 is *not* capable of Interrupt Handler Control.

Interrupt Status Reporting Bit 0*

"**0**" indicates that the V535 has Interrupt Status Reporting capability.

Serial Number Register *0Ah, 0Ch*

The read-only Serial Number registers (high and low words) store the 32-bit hexadecimal value of the V535's decimal serial number.

0Ah

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	SN7				SN6				SN5				SN4			

High word

0Ch

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	SN3				SN2				SN1				SN0			

Low word

Version Number Register *0Eh*

This read-only register gives the hardware and firmware revision numbers of the module.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	Firmware Version				Firmware Revision				Hardware Version				Hardware Revision			

Firmware Version	***Bits 15 - 12***
Firmware Revision	***Bits 11 - 8***
Hardware Version	***Bits 7 - 4***
Hardware Revision	***Bits 3 - 0***

Each field is a four-bit integer indicating the version or revision number.

Interrupt Status Register 1Ah

This read-only register provides information about the state of the interrupt sources within the V535. The Status field of this register is configured for the two groups of 4 axis controllers. Bits 8 through 11 are for axis' 1 through 4, and bits 12 through 15 are for axes 5 through 8. The Interrupt Status Register can be used for polling of interrupts. Interrupt bits are cleared by a read of this register or by an interrupt-acknowledge cycle.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	Aux Digital Input B	Seq Done B	SMIC Ready B	SMIC IRQ B	Aux Digital Input A	Seq Done A	SMIC Ready A	SMIC IRQ A	Logical Address							
	Axis 5-8 Controller				Axis 1-4 Controller											

Auxiliary Digital Input Group B Bit 15

A "1" read back from this bit location indicates that the preselected digital input has been asserted for Group B. Group B refers to axes 8 through 5 on the V535. This interrupt source is edge sensitive. Once the interrupt source is generated from this condition, it must be cleared or masked off to allow subsequent interrupt sources from this location. This bit is auto-cleared when this register is read or an interrupt acknowledge cycle is executed.

Stepper Motor Sequencer Done Group B Bit 14

A "1" read back from this bit location indicates that the Group B (Axes 8 through 5) sequencer command has completed. This bit is latched and auto-cleared when this register is read or an interrupt acknowledge cycle is executed.

Stepper Motor Interface Controller Ready Group B Bit 13

A "1" read back from this bit location indicates that the READY signal from the Stepper Motor Interface Controller (SMIC) has been asserted. This READY bit is set false when the SMIC is busy executing an operation and is then set true when complete. This status can then be used to generate an interrupt. This bit is latched and auto-cleared when this register is read or an interrupt acknowledge cycle is executed.

Stepper Motor Interface Controller Interrupt Request Group B Bit 12

A "1" is read back for this bit location when the Stepper Motor Interface Controller (SMIC) has an interrupt pending. This can be the result of many sources within the SMIC and must be queried by command to verify the actual source. This interrupt source is the logical AND of the interrupt enable bit for this source and the interrupt signal from the SMIC. Once an interrupt has been acknowledged and serviced by the host computer, the interrupt source enable bit should be re-enabled. If another interrupt source inside the SMIC has occurred since last source, the re-enabling will generate an immediate interrupt. This bit is auto-cleared. To remove the source of the interrupt, the SMIC must be serviced appropriately.

Auxiliary Digital Input Group A Bit 11

A "1" read back from this bit location indicates that the preselected digital input has been asserted for Group A. Group A refers to axes 4 through 1 on the V535. This interrupt source is edge sensitive. Once the interrupt source is generated from this condition, it must be cleared or masked off to allow subsequent interrupt sources from this location. This bit is auto-cleared when this register is read or an interrupt acknowledge cycle is executed.

Stepper Motor Sequencer Done Group A *Bit 10*

A "1" read back from this bit location indicates that the Group B (Axes 4 through 1) sequencer command has completed. This bit is latched and auto-cleared when this register is read or an interrupt acknowledge cycle is executed.

Stepper Motor Interface Controller Ready Group A *Bit 9*

A "1" read back from this bit location indicates that the READY signal from the Stepper Motor Interface Controller (SMIC) has been asserted. This READY bit is set false when the SMIC is busy executing an operation and is then set true when complete. This status can then be used to generate an interrupt. This bit is latched and auto-cleared when this register is read or an interrupt acknowledge cycle is executed.

Stepper Motor Interface Controller Interrupt Request Group A *Bit 8*

A "1" is read back for this bit location when the Stepper Motor Interface Controller (SMIC) has an interrupt pending. This can be the result of many sources within the SMIC and must be queried by command to verify the actual source. This interrupt source is the logical AND of the interrupt enable bit for this source and the interrupt signal from the SMIC. Once an interrupt has been acknowledged and serviced by the host computer, the interrupt source enable bit should be re-enabled. If another interrupt source inside the SMIC has occurred since last source, the re-enabling will generate an immediate interrupt. This bit is auto-cleared. To remove the source of the interrupt, the SMIC must be serviced appropriately.

Logical Address *Bits 7 - 0*

During a programmed control read operation, these bits return all "1's." During an interrupt acknowledge cycle, these bits return the V535's Logical Address.

Interrupt Control Register *1Ch*

The read/write register contains mask bits for the interrupt source, a bit for disabling interrupts and three bits that determine interrupt level. All of the bits in this register are set to "1" on the assertion of SYSRESET.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/write	Aux Digital Input B	Seq Done B	SMIC Ready B	SMIC IRQ B	Aux Digital Input A	Seq Done A	SMIC Ready A	SMIC IRQ A	EN*	1	Interrupt Req. Level			1	1	1

Auxiliary Digital Input Group B Interrupt Mask Bit *Bit 15*

Writing a "1" to this bit *prevents* the Auxiliary Digital Input from Group B (Axes 8 through 5) interrupt source from generating an interrupt request. Writing a **"0"** enables an interrupt source to generate an interrupt request.

Stepper Motor Sequencer Done Group B Interrupt Mask Bit Bit 14

Writing a "1" to this bit *prevents* the Stepper Motor Sequencer Done Group B (Axes 8 through 5) interrupt source from generating an interrupt request. Writing a "0" enables an interrupt source to generate an interrupt request.

Stepper Motor Interface Controller Ready Group B Interrupt Mask Bit Bit 13

Writing a "1" to this bit *prevents* the Stepper Motor Interface Controller Ready Group B (Axes 8 through 5) interrupt source from generating an interrupt request. Writing a "0" enables an interrupt source to generate an interrupt request.

Stepper Motor Interface Controller Interrupt Request Group B Interrupt Mask Bit 12

Writing a "1" to this bit *prevents* the Stepper Motor Interface Controller Interrupt Request Group B (Axes 8 through 5) interrupt source from generating an interrupt request. Writing a "0" enables an interrupt source to generate an interrupt request.

Auxiliary Digital Input Group A Interrupt Mask Bit Bit 11

Writing a "1" to this bit *prevents* the Auxiliary Digital Input from Group A (Axes 4 through 1) interrupt source from generating an interrupt request. Writing a "0" enables an interrupt source to generate an interrupt request.

Stepper Motor Sequencer Done Group A Interrupt Mask Bit Bit 10

Writing a "1" to this bit *prevents* the Stepper Motor Sequencer Done Group A (Axes through 1 interrupt source from generating an interrupt request. Writing a "0" enables an interrupt source to generate an interrupt request.

Stepper Motor Interface Controller Ready Group A Interrupt Mask Bit Bit 9

Writing a "1" to this bit *prevents* the Stepper Motor Interface Controller Ready Group A (Axes 4 through 1 interrupt source from generating an interrupt request. Writing a "0" enables an interrupt source to generate an interrupt request.

Stepper Motor Interface Controller Interrupt Request Group A Interrupt Mask Bit 8

Writing a "1" to this bit *prevents* the Stepper Motor Interface Controller Interrupt Request Group A (Axes 4 through 1) interrupt source from generating an interrupt request. Writing a "0" enables an interrupt source to generate an interrupt request.

Interrupt Enable Bit 7

Writing a "1" to this bit disables interrupt generation. Writing a "0" enables interrupt generation.

Interrupt Request Level *Bits 5 - 3*

These bits determine the interrupt request level.

Bits			Interrupt Request Level
5	4	3	
0	0	0	IRQ7
0	0	1	IRQ6
0	1	0	IRQ5
0	1	1	IRQ4
1	0	0	IRQ3
1	0	1	IRQ2
1	1	0	IRQ1
1	1	1	Disconnected

Table 4-2: Interrupt Request Levels

Not Used Bits 2 – 0

Bits 2 – 0 are not used and return **"1"**s when read.

Subclass Register *1Eh*

This read-only register provides information about the Subclass of the VXIbus device.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	Extended Device	Extended Register-based Device														

Bit 15 indicates that the V535 is a VXIbus-defined Extended Device.

Bits 14 through 0 indicate that the V535 is an Extended Register-based Device.

Suffix Register 20h, 22h

The Suffix read-only register (high and low words) holds the ASCII codes for the four characters of the V535's suffix. The suffix defines the optional characteristics of the module.

20h

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	0	1	0	0	1	1	0	0	0	1	0	0	0	0	x	x

ASCII code for L = 4Ch (1st character) **ASCII code for A or B = 41h or 42h (2nd character)**

22h

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read-only	0	0	1	1	0	0	x	x	0	0	1	1	0	0	x	x

ASCII code for 1 or 2 = 31h or 32h (3rd character) **ASCII code for 1, 2, etc. = 31h, 32h, etc. (4th character)**

The suffix options for the V535 are:

1st character: "**L**" for all options

2nd character: Encode Option

"**A**" indicates no encoder.

"**B**" indicates optional encoder.

3rd character: Number of Axis

"**1**" indicates a module with 4 axis.

"**2**" indicates a module with 8 axis.

4th character: Revision level

The number, "**1**," "**2**," etc., gives the module revision level.

Operational Registers in A24 Space

The following operational registers are in **A24** space beginning at a starting address specified by the Offset register in **A16** configuration space. The hexadecimal addresses shown are the offset from the starting address. You can access these registers by **D16** or **D32** transfers. Any register bit that is shown as "**0**" is always read as "**0**" regardless of the data that is written to it. *For the 4-axis versions of this module, only the bits representing Axes 4 to 1 are used.*

Group A Command/Status Register *00h*

This read/write register is used to initiate stepper indexer operations and monitor its status for Stepper Motor Axes 4 through 1. These four axes are considered to be part of Group A.

A write operation to this register initiates a data transfer operation to the Stepper Motor Indexer Controller for Group A. Group A consists of Axes 4 through 1 on the module. Once this register is written with the appropriate data specifying the type of operation to execute, it may then be read to monitor the status of the requested operation. Please refer to the following bit descriptions for this register and the section entitled "*Executing Stepper Motor Indexer Commands*".

Read/Write	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Direction	Word Count 1	Word Count 0	Checksum Enable	IOC Busy	Sequence Busy	Transfer Mode 1	Transfer Mode 0	Command							

For **D16** transfers, Bits 15-0 are accessed at **02h**.

Direction *Bit 15*

This is a write/read bit used to specify the direction of the transfer to/from the Stepper Motor Indexer Controller (SMIC). Set this bit to a "**1**" to specify a READ from the SMIC and clear this bit to a "**0**" to specify a WRITE to the SMIC. The DIRECTION bit is used in combination with the WORD COUNT bits to define the number of 16-bit data words to be moved to/from the SMIC. The DIRECTION bit is ignored if an operation is executed and both the WORD COUNT bits are set to zero. A Word Count of zero indicates that the operation to the SMIC requires only the Command field and no data.

Word Count 1, Word Count 0 *Bits 14-13*

These two write/read bits are used to specify the number of 16-bit data words that are to be transferred to or from the SMIC during the requested operation. This value does not include any data words transferred for the checksum. The data word transfer for the checksum is either enabled or disabled through the CHECKSUM ENABLE bit in this register.

The binary combination of the two WORD COUNT bits determine the number of 16-bits words to be transferred as follows:

Word Count 1	Word Count 0	Number of 16-Bit Data Words to Transfer
0	0	0
0	1	1
1	0	2
1	1	Reserved

Table 4-3: Selection values for Word Count Specification

During a write operation to the SMIC, the write data is contained in the Group A Data Register. The Group A Data Register is a 32-bit data register that can hold the maximum two 16-bit data words for transfer. If the WORD COUNT specifies a count of zero, no data is transferred from the Group A Data Register. For WORD COUNT specifications of one, only the lower 16-bits (Bits 15-0) of the Group A Data Register are used. WORD COUNT specifications of two use the entire contents of the Group A Data Register.

During a read operation to the SMIC, the read data is returned in the Group A Data Register. The Group A Data Register is a 32-bit data register that can hold the maximum two 16-bit data words received from the data transfer to the SMIC. If the WORD COUNT specifies a count of zero, no data is retrieved from the SMIC. For WORD COUNT specifications of one, the 16-bits of data retrieved from the SMIC is stored into the lower 16-bits (Bits 15-0) of the Group A Data Register. WORD COUNT specifications of two use the entire 32-bit field of the Group A Data Register.

Checksum Enable *Bit 12*

This write/read bit is used to enable and disable the retrieval of the Checksum value from the SMIC. Set this bit to a **"1"** to enable the checksum and a **"0"** to disable the checksum. The Checksum value consists of a 16-bit sum of all previous communications that have occurred for the associated command. To calculate the checksum, add all data values transferred to/from the SMIC during a command operation. Since the checksum is a 16-bit value, remove any data bits past the first 16. This calculated value can then be compared to the returned checksum from the SMIC. When the checksum is enabled, the checksum value returned is stored in the Group A Checksum Data Register.

Reading the Checksum value from the SMIC is optional. Recovering from a bad checksum will depend on the nature of the command. Read and Write operations to the SMIC can always be repeated, while a command resulting in a specific action may or may not be retried, depending on the command executed and the state of the axis at the time the command was executed.

IOC Busy *Bit 11*

I/O Controller Busy is a read-only bit that is read back as a **"1"** when the I/O Controller is busy executing a requested operation. Once the I/O Controller has completed the requested operation, this bit is read back as a **"0"**.

Before any write operation can occur to the Group A Command/Status Register to perform an operation, this bit must be read back as a **"0"**. The I/O Controller ***cannot*** be instructed to execute any commands while it is busy. Failure to observe this rule can result in indeterminate behavior.

Optionally, an interrupt can be generated when the IOC Busy bit makes the transition from a "1" to a "0". This transition is an indication that the command or data word was successfully transferred to/from the SMIC. Using this interrupt mechanism can reduce system software overhead since polling of the Group A Command/Status Register would not be required.

Sequence Busy *Bit 10*

Sequence Busy is a read-only bit that is read back as a "1" as long as the Command Sequencer is busy executing an operation. Once the requested operation is complete, this bit is read back as a "0".

Before any write operation can occur to the Group A Command/Status Register to perform an operation, this bit must be read back as a "0". The Command Sequencer ***cannot*** be instructed to execute any commands while it is busy. Failure to observe this rule can result in indeterminate behavior.

Optionally, an interrupt can be generated when the Sequencer Busy bit makes the transition from a "1" to a "0". This transition is an indication that the requested sequencer operation was completed. Using this interrupt mechanism can reduce system software overhead since polling of the Group A Command/Status Register would not be required.

Transfer Mode 1, Transfer Mode 0 *Bits 9-8*

The Transfer Mode 1 and Transfer Mode 0 bits are write/read bits used to specify the mode of operation for the Command Sequencer. The Command Sequencer can be set to operate in one of three modes. The first mode, termed Sequencer Mode, removes the burden from the host processor during command sequences in which polling for the IOC BUSY bit is required. Polling of the IOC BUSY bit is required before storage of the Command byte into the SMIC, before storage of Data bytes into the SMIC and before retrieval of Data bytes from the SMIC. The Sequencer Mode performs the polling operations automatically and eliminates the number of polling operations or interrupt cycles required to execute operations.

The second mode, termed Individual Command Mode, provides a very basic interface for sending Command bytes to the SMIC. This mode is used only for transferring of Command bytes and places the burden of polling the IOC Busy bit on the host computer. Use of this mode of operation permits direct access to the SMIC without the intervention of the Command Sequencer.

The third mode, termed Individual Data Mode, provides a very basic interface for sending/receiving Data bytes to/from the SMIC. This mode is used only for the transfer of Data bytes associated with a Command, not the transfer of the Command byte itself and places the burden of polling the IOC Busy bit on the host computer. Use of this mode permits direct access to the SMIC without the intervention of the Command Sequencer.

The binary combination of the Transfer Mode bits determines which mode is selected according to the following table.

Transfer Mode 1	Transfer Mode 0	Transfer Mode Selected
0	0	Sequencer Mode
0	1	Individual Command Mode
1	0	Individual Data Mode
1	1	Reserved

Table 4-4: Selection values for Sequencer Mode Specification

Please refer to the section entitled "*Executing Stepper Motor Indexer Commands*" for additional details on the various Transfer Modes.

Command ***Bits 7-0***

The Command bits write/read bits are used to specify the Command byte to be sent to the SMIC during either Sequencer Mode or Individual Data Mode operations. The data loaded into these bit locations is sent to the SMIC during the requested operation.

Group A Data Register *04h*

This read/write register is used to hold data values for transfer to/from the SMIC. During SMIC write operations, the data located in this register is written to the SMIC. During SMIC read operations, the data retrieved from the SMIC is loaded into this register. This register is used for Axes 4 through 1 only.

The operation of this register varies depending on the Transfer Mode, Word Count and the Direction selected in the Group A Command/Status Register. During Sequencer Mode SMIC write operations, the data contained in this register is moved to the SMIC if the Word Count specification is not zero. The number of 16-bit words moved to the SMIC is defined by the Word Count. If the Word Count specifies one word, bits 15 through 0 of the Group A Data Register are written to the SMIC after the Command byte has transferred. If the Word Count specifies a count of two, bits 15 through 0 of the Group A Data Register are written to the SMIC after the Command byte has transferred, followed by bits 31 through 16.

During Sequencer Mode SMIC read operations, the data retrieved from the read of the SMIC is stored in this register. The Command Sequencer retrieves data from the SMIC after a Command byte is transferred and the Word Count specification is not zero. If the Word Count specifies a count of one, bits 15 through 0 of the Group A Data Register are updated with the 16-bit data word received from the SMIC. If the Word Count specifies a count of two, bits 15 through 0 of the Group A Data Register are updated with the first 16-bit data word received from the SMIC and then bits 31 through 16 are updated with the second 16-bit word received from the SMIC.

The Group A Data Register is also used for the Individual Data Transfer Modes of operation. For Individual Data Mode SMIC write operations, the data contained in bit locations 7 through 0 are loaded into the SMIC. For Individual Data Mode SMIC read operations, the data received from the SMIC is loaded into but locations 7 through 0 of the Group A Data Register. Since Individual Data Transfer Modes only transfer 8-bits of data, Group A Data Register bits 31 through 8 are not used.

The following shows the bit layout for the Group A Data Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read/Write	W/R 31	W/R 30	W/R 29	W/R 28	W/R 27	W/R 26	W/R 25	W/R 24	W/R 23	W/R 22	W/R 21	W/R 20	W/R 19	W/R 18	W/R 17	W/R 16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	W/R 15	W/R 14	W/R 13	W/R 12	W/R 11	W/R 10	W/R 9	W/R 8	W/R 7	W/R 6	W/R 5	W/R 4	W/R 3	W/R 2	W/R 1	W/R 0

For **D16** transfers, Bits 31 – 16 are accessed at **04h** and Bits 15-0 are accessed at **06h**.

Group A Checksum Register 08h

This read-only register is used to hold the Checksum value returned from a checksum read command from the SMIC. The Checksum value consists of a 16-bit sum of all previous communications that have occurred for the associated command. To calculate the checksum, add all data values transferred to/from the SMIC during a command operation. Since the checksum is a 16-bit value, remove any data bits past the first 16. This calculated value can then be compared to the returned checksum from the SMIC. When the checksum is enabled, the checksum value returned is stored in the Group A Checksum Data Register.

Reading the Checksum value from the SMIC is optional. The Checksum is enabled by setting the Checksum Enable bit in the Group A Command/Status Register to a "**1**". The Checksum is disabled by setting the Checksum Enable bit to a "**0**".

The following diagram shows the bit pattern for the Group A Checksum Register.

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

For **D16** transfers, Bits 15-0 are accessed at **0Ah**.

Group B Command/Status Register 0Ch+

This read/write register is used to initiate stepper indexer operations and monitor its status for Stepper Motor Axes 8 through 4. These four axes are considered to be part of Group B. If the option of the V535 only includes 4 axes, the sections of the manual concerning Group B Registers can be ignored.

A write operation to this register initiates a data transfer operation to the Stepper Motor Indexer Controller for Group B. Group B consists of Axes 8 through 4 on the module. Once this register is written with the appropriate data specifying the type of operation to execute, the register may then be read to monitor the status of the requested operation. Please refer to the following bit descriptions for this register and the section entitled "*Executing Stepper Motor Indexer Commands*".

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read/Write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/Write	Direction	Word Count 1	Word Count 0	Check-sum Enable	IOC Busy	Sequence Busy	Transfer Mode 1	Transfer Mode 0	Command							

For **D16** transfers, Bits 15-0 are accessed at **0Eh**.

Direction *Bit 15*

This is a write/read bit used to specify the direction of the transfer to/from the Stepper Motor Indexer Controller (SMIC). Set this bit to a **"1"** to specify a READ from the SMIC and clear this bit to a **"0"** to specify a WRITE to the SMIC. The DIRECTION bit is used in combination with the WORD COUNT bits to define the number of 16-bit data words to be moved to/from the SMIC. The DIRECTION bit is ignored if an operation is executed and both the WORD COUNT bits are set to zero. A Word Count of zero indicates that the operation to the SMIC requires only the Command field and no data.

Word Count 1, Word Count 0 *Bits 14-13*

These two write/read bits are used to specify the number of 16-bit data words that are to be transferred to or from the SMIC during the requested operation. This value does not include any data words transferred for the checksum. The data word transfer for the checksum is either enabled or disabled through the CHECKSUM ENABLE bit in this register.

The binary combination of the two WORD COUNT bits determine the number of 16-bits words to be transferred as follows:

Word Count 1	Word Count 0	Number of 16-Bit Data Words to Transfer
0	0	0
0	1	1
1	0	2
1	1	Reserved

Table 4-5: Selection values for Word Count Specification

During a write operation to the SMIC, the write data is contained in the Group B Data Register. The Group B Data Register is a 32-bit data register that can hold the maximum two 16-bit data words for transfer. If the WORD COUNT specifies a count of zero, no data is transferred from the Group B Data Register For WORD COUNT specifications of one, only the lower 16-bits (Bits 15-0) of the Group B Data Register are used. WORD COUNT specifications of two use the entire contents of the Group B Data Register.

During a read operation to the SMIC, the read data is returned in the Group B Data Register. The Group B Data Register is a 32-bit data register that can hold the maximum two 16-bit data words received from the data transfer from the SMIC. If the WORD COUNT specifies a count of zero, no data is retrieved from the SMIC. For WORD COUNT specifications of one, the 16-bits of data retrieved from the SMIC are stored into the lower 16-bits (Bits 15-0) of the Group B Data Register. WORD COUNT specifications of two use the entire 32-bit field of the Group B Data Register. The first 16-bit data word retrieved from the SMIC is stored in the lower 16-bits of the Group B Data Register and the second 16-bit word is stored in the upper 16-bits of the register (Bits 31-16).

Checksum Enable *Bit 12*

This write/read bit is used to enable and disable the retrieval of the Checksum value from the SMIC. Set this bit to a **"1"** to enable the checksum and a **"0"** to disable the checksum. The Checksum value consists of a 16-bit sum of all previous communications that have occurred for the associated command. To calculate the checksum, add all data values transferred to/from the SMIC during a command operation. Since the checksum is a 16-bit value, remove any data bits past the first 16. This calculated value can then be compared to the returned checksum from the SMIC. When the checksum is enabled, the checksum value returned is stored in the Group B Checksum Data Register.

Reading the Checksum value from the SMIC is optional. Recovering from a bad checksum will depend on the nature of the command. Read and Write operations to the SMIC can always be repeated, while a command resulting in a specific action may or may not be retried, depending on the command executed and the state of the axis at the time the command was executed.

IOC Busy Bit 11

I/O Controller Busy is a read-only bit that is read back as a "1" when the I/O Controller is busy executing a requested operation. Once the I/O Controller has completed the requested operation, this bit is read back as a "0".

Before any write operation can occur to the Group A Command/Status Register to perform an operation, this bit must be read back as a "0". The I/O Controller ***cannot*** be instructed to execute any commands while it is busy. Failure to observe this rule can result in indeterminate behavior.

Optionally, an interrupt can be generated when the IOC Busy bit makes the transition from a "1" to a "0". This transition is an indication that the command or data word was successfully transferred to/from the SMIC. Using this interrupt mechanism can reduce system software overhead since polling of the Group B Command/Status Register would not be required.

Sequence Busy Bit 10

Sequence Busy is a read-only bit that is read back as a "1" as long as the Command Sequencer is busy executing an operation. Once the requested operation is complete, this bit is read back as a "0".

Before any write operation can occur to the Group A Command/Status Register to perform an operation, this bit must be read back as a "0". The Command Sequencer ***cannot*** be instructed to execute any commands while it is busy. Failure to observe this rule can result in indeterminate behavior.

Optionally, an interrupt can be generated when the Sequencer Busy bit makes the transition from a "1" to a "0". This transition is an indication that the requested sequencer operation was completed. Using this interrupt mechanism can reduce system software overhead since polling of the Group B Command/Status Register would not be required.

Transfer Mode 1, Transfer Mode 0 Bits 9-8

The Transfer Mode 1 and Transfer Mode 0 bits are write/read bits used to specify the mode of operation for the Command Sequencer. The Command Sequencer can be set to operate in one of three modes. The first mode, termed Sequencer Mode, removes the burden from the host processor during command sequences in which polling for the IOC BUSY bit is required. Polling of the IOC BUSY bit is required before storage of the Command byte into the SMIC, before storage of Data bytes into the SMIC and before retrieval of Data bytes from the SMIC. The Sequencer Mode performs the polling operations automatically and eliminates the number of polling operations or interrupt cycles required to execute operations.

The second mode, termed Individual Command Mode, provides a very basic interface for sending Command bytes to the SMIC. This mode is used only for transferring of Command bytes and places the burden of polling the IOC Busy bit on the host computer. Use of this mode of operation permits direct access to the SMIC without the intervention of the Command Sequencer.

The third mode, termed Individual Data Mode, provides a very basic interface for sending/receiving Data bytes to/from the SMIC. This mode is used only for the transfer of Data bytes associated with a Command, not the transfer of the Command byte itself and places the burden of polling the IOC Busy bit on the host computer. Use of this mode permits direct access to the SMIC without the intervention of the Command Sequencer.

The binary combination of the Transfer Mode bits determine which mode is selected according to the following table:

Transfer Mode 1	Transfer Mode 0	Transfer Mode Selected
0	0	Sequencer Mode
0	1	Individual Command Mode
1	0	Individual Data Mode
1	1	Reserved

Table 4-6: Selection values for Sequencer Mode Specification

Please refer to the section entitled "*Executing Stepper Motor Indexer Commands*" for additional details on the various Transfer Modes.

Command ***Bits 7-0***

The Command bits write/read bits are used to specify the Command byte to be sent to the SMIC during either Sequencer Mode or Individual Data Mode operations. The data loaded into these bit locations is sent to the SMIC during the requested operation.

Group B Data Register *10h*

This read/write register is used to hold data values for transfer to/from the SMIC. During SMIC write operations, the data located in this register is written to the SMIC. During SMIC read operations, the data retrieved from the SMIC is loaded into this register. This register is used for Axes 4 through 7 only.

The operation of this register varies depending on the Transfer Mode, Word Count and the Direction selected in the Group B Command/Status Register. During Sequencer Mode SMIC write operations, the data contained in this register is moved to the SMIC if the Word Count specification is not zero. The number of 16-bit words moved to the SMIC is defined by the Word Count. If the Word Count specifies one word, bits 15 through 0 of the Group A Data Register are written to the SMIC after the Command byte has transferred. If the Word Count specifies a count of two, bits 15 through 0 of the Group B Data Register are written to the SMIC after the Command byte has transferred, followed by bits 31 through 16.

During Sequencer Mode SMIC read operations, the data retrieved from the read of the SMIC is stored in this register. The Command Sequencer retrieves data from the SMIC after a Command byte is transferred and the Word Count specification is not zero. If the Word Count specifies a count of one, bits 15 through 0 of the Group B Data Register are updated with the 16-bit data word received from the SMIC. If the Word Count specifies a count of two, bits 15 through 0 of the Group B Data Register are updated with the first 16-bit data word received from the SMIC and then bits 31 through 16 are updated with the second 16-bit word received from the SMIC.

The Group B Data Register is also used for the Individual Data Transfer Modes of operation. For Individual Data Mode SMIC write operations, the data contained in bit locations 7 through 0 are loaded into the SMIC. For Individual Data Mode SMIC read operations, the data received from the SMIC is loaded into but locations 7 through 0 of the Group B Data Register. Since Individual Data Transfer Modes only transfer 8-bits of data, Group B Data Register bits 31 through 8 are not used.

The following shows the bit layout for the Group B Data Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read/write	W/R 31	W/R 30	W/R 29	W/R 28	W/R 27	W/R 26	W/R 25	W/R 24	W/R 23	W/R 22	W/R 21	W/R 20	W/R 19	W/R 18	W/R 17	W/R 16

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	W/R 15	W/R 14	W/R 13	W/R 12	W/R 11	W/R 10	W/R 9	W/R 8	W/R 7	W/R 6	W/R 5	W/R 4	W/R 3	W/R 2	W/R 1	W/R 0

For **D16** transfers, Bits 31-16 are accessed at **10h** and Bits 15-0 are accessed at **012h**.

Group B Checksum Register 14h

This read-only register is used to hold the Checksum value returned from a checksum read command from the SMIC. The Checksum value consists of a 16-bit sum of all previous communications that have occurred for the associated command. To calculate the checksum, add all data values transferred to/from the SMIC during a command operation. Since the checksum is a 16-bit value, remove any data bits past the first 16. This calculated value can then be compared to the returned checksum from the SMIC. When the checksum is enabled, the checksum value returned is stored in the Group B Checksum Data Register.

Reading the Checksum value from the SMIC is optional. The Checksum is enabled by setting the Checksum Enable bit in the Group B Command/Status Register to a **"1"**. The Checksum is disabled by setting the Checksum Enable bit to a **"0"**.

The following diagram shows the bit pattern for the Group B Checksum Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read-Only	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CKSM 15	CKSM 14	CKSM 13	CKSM 12	CKSM 11	CKSM 10	CKSM 9	CKSM 8	CKSM 7	CKSM 6	CKSM 5	CKSM 4	CKSM 3	CKSM 2	CKSM 1	CKSM 0

For **D16** transfers, Bits 15-0 are accessed at **16h**.

Multi-Axis Command Register 18h

This write/read register can be used to send a command to multiple axes without host computer intervention. These commands can only be SMIC commands for which there is no required data transfer. This mechanism provides a quick way to update several (all) axes with the same command.

This register consists of an 8-bit field representing the command to be executed for each axis. Another set of 8 bits is used to indicate for which axis the command is to be executed. A hardware facility on the V535 provides the individual commands to each SMIC to switch axis. The command specified in this register is then supplied to each axis that has its corresponding bit set to a "1". Any bit set to a "0" when the register is written will remain unchanged.

Once the register is written, the hardware processor starts sending the appropriate commands to each SMIC. The hardware processor starts at axis 1 for each group and continues until axis 4 for the group. Any bit set to a "1" causes the axis to be selected and the command executed. Once the selected command is executed to the indicated axis, its corresponding bit is then cleared to "0". The operation(s) are considered complete once all bits have been reset to "0" by the hardware processor.

Also, the Sequence Busy bit for each Group is set to a "1" until all axes specified in the field for each group are reset to "0". Therefore, one can use one of four mechanisms to check for completion of the operation.

1.) Poll the Multi-Axis Command Register until all bits are read back as "0".

2.) Poll the Group A and Group B Command/Status Registers until both Sequence Busy bits are read back as "0".

3.) Poll the Interrupt Status Register until both Sequence Busy bits are read back as "0".

4.) Enable interrupts so that an interrupt will be generated to the host computer when each group has finished processing the commands.

Note: Care should be taken when using this command to ensure a command is not specified that requires the transfer of data other than the Command byte. Also, after the requested multi-axis command operation is complete, the active axis for each group will be the axis with the most significant setting in the register. For example, if a multi-axis command were to be executed to axis 1 and 3, axis 3 would be the selected axis after the command completes.

The following shows the bit pattern for the Multi-Axis Command Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Write/Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Command								Axis 8	Axis 7	Axis 6	Axis 5	Axis 4	Axis 3	Axis 2	Axis 1
									Group B				Group A			

For **D16** transfers, Bits 15-0 are accessed at **1Ah**.

Command Bits 15-8

The Command bits are write/read bits that are used to specify the Multi-Axis command to be executed. This command cannot include operations that require data transfers, only command byte transfers.

Group B Axis 8-5 Bits 7-4

These write/read bits are used to specify which axes in Group B are to have the command executed. Any bit set to a "**1**" when this register is written requests that the command be executed to that axis. Any bit written to a "**0**" will be exempt from having the command executed on the corresponding axis. A read of these bits indicates when the requested operation is complete. Any bit read back as a "**0**"indicates that the operation is complete on that axis. Any bit read back as a "**1**" indicates that the operation is still pending for the corresponding axis.

Group A Axis 4-1 Bits 3-0

These write/read bits are used to specify which axis in Group A are to have the command executed. Any bit set to a "**1**" when this register is written requests that the command be executed to that axis. Any bit written to a "**0**" will be exempt from having the command executed on the corresponding axis. A read of these bits indicates when the requested operation is complete. Any bit read back as a "**0**"indicates that the operation is complete on that axis. Any bit read back as a "**1**" indicates that the operation is still pending for the corresponding axis.

Digital I/O Control Register 1Ch

This write/read register is used to specify the level of debouncing that the limit inputs to the V535 will utilize. Along with the debouncer setup, this register also contains several bits for determining the input polarity of the limit inputs and the type of control signals to use for interfacing to the stepper motor translator.

The debouncing configuration ensures that noise on any of the limit inputs is not used to falsely trigger any of the limits. The level of debouncing ranges from 2 milliseconds to 32 milliseconds in a binary progression. For the selected debounce rate, the limit input to the V535 must be valid for the selected amount of time before the input is recognized. The selection for the debouncers covers all limit inputs.

The limit inputs to the V535 consist of two per axis, a Clockwise (CW) limit and a Counter-Clockwise (CCW) limit. The limit inputs are used to inform the SMIC that a travel limit has been reached. The host can be automatically notified of a limit condition by generating an interrupt. The SMIC interrupt from each group can generate its own individual interrupt to VXI. This interrupt is enable/disabled through the Interrupt Mask Register located in A16 address space. Once a SMIC interrupt is generated, host software must interrogate the SMIC in order to determine which axis generated the limit.

In order to reduce the effect of a noisy environment on the sensing of limit switch closures, the V535 implements a nominal 10-milliampere isolated current loop. Please refer to the *"Limit and Auxiliary Digital Inputs"* section of this manual for additional information.

The following diagram shows the bit pattern for the Limit Input Debounce Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Write/Read	Axis8 CCW Limit Polarity	Axis8 CW Limit Polarity	Axis7 CCW Limit Polarity	Axis7 CW Limit Polarity	Axis6 CCW Limit Polarity	Axis6 CW Limit Polarity	Axis5 CCW Limit Polarity	Axis5 CW Limit Polarity	Axis4 CCW Limit Polarity	Axis4 CW Limit Polarity	Axis3 CCW Limit Polarity	Axis3 CW Limit Polarity	Axis2 CCW Limit Polarity	Axis2 CW Limit Polarity	Axis1 CCW Limit Polarity	Axis1 CW Limit Polarity
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Axis8-5 Pulse Pol	Axis8-5 Pulse	Axis8-5 Dir Pol	Axis4-1 Pulse Pol	Axis4-1 Pulse	Axis4-1 Dir Pol	Axis8-5 Pulse Config	Axis4-1 Pulse Config	0	0	0	0	0	DBNC 2	DBNC 1	DBNC 0

For **D16** transfers, Bits 15-0 are accessed at **1Eh**.

Axis 8-1 CCW/CW Limit Polarity *Bits 31-16*

These write/read bits are used to specify the polarity of the limit inputs to the V535. All sixteen bits are used for an 8 axis V535 and only bits 23 through 16 are used for a 4 axis V535. Setting any of these bits to a **"0"** indicates that the input is to be recognized as high true and no inversion of the input signal is introduced. A **"1"** in any bit location indicates that the limit input is to be seen as low true and the input signal is inverted in order to preserve its logical sense.

Axis 8-5 Pulse Polarity *Bit 15*

This write/read bit is used to specify the polarity of the pulse signal for axis 8 through 5. Setting this bit to a "0" indicates that the pulse output is to be high true and no inversion of the signal is introduced. Setting this bit to a "1" indicates that the pulse output is to be recognized as low true and the signal is inverted.

Axis 8-5 Pulse *Bit 14*

This write/read bit is used to specify a one shot extension on the pulse signal on the output. The width of the pulse is 8 microseconds.

Axis 8-5 Direction Polarity *Bit 13*

This write/read bit is used to specify the polarity of the direction signal for axis 8 through 5. Setting this bit to a "0" indicates the direction output is to be high true and no inversion of the signal is introduced. Setting this bit to a "1" indicates the direction output is to be low true and the signal is inverted.

Axis 4-1 Pulse Polarity *Bit 12*

This write/read bit is used to specify the polarity of the pulse signal for axis 4-1. Setting this bit to a "0" indicates that the pulse output is to be high true and no inversion of the signal is introduced. Setting this bit to a "1" indicates that the pulse output is to be low true and the input signal is inverted.

Axis 4-1 Pulse *Bit 11*

This write/read bit is used to specify a one shot extension on the pulse signal on the output. The width of the pulse is 8 microseconds.

Axis 4-1 Direction Polarity *Bit 10*

This write/read bit is used to specify the polarity of the direction signal for axis 4 through 1. Setting this bit to a "0" indicates the direction output is to be high true and no inversion of the signal is introduced. Setting this bit to a "1" indicates the direction output is to be low true and the signal is inverted.

Axis 8-5 Pulse Configuration *Bit 9*

These write/read bits are used to specify the output signal convention to be used by the V535. This bit is used to control the configuration for axis 8 through 5. A "**0**" in this bit location specifies that the V535 should use the PULSE and DIRECTION convention for generating signals to the stepper motor translator. A "**1**" in this bit location specifies the use of the CLOCKWISE and COUNTERCLOCKWISE signal convention.

Axis 4-1 Pulse Configuration *Bit 8*

These write/read bits are used to specify the output signal convention to be used by the V535. This bit is used to control the configuration for axis 1 through 4. A "**0**" in this bit location specifies that the V535 should use the PULSE and DIRECTION convention for generating signals to the stepper motor translator. A "**1**" in this bit location specifies the use of the CLOCKWISE and COUNTERCLOCKWISE signal convention.

Debounce 2 - 0 *Bits 2-0*

These write/read bits are used to set up the debounce time for the limit inputs. The binary combination of these three bits setup the debounce rate as shown in the following table.

DBNC2	DBNC1	DBNC0	Debounce Time Selection
0	0	0	2 Milliseconds
0	0	1	4 Milliseconds
0	1	0	8 Milliseconds
0	1	1	16 Milliseconds
1	0	0	32 Milliseconds
1	0	1	32 Milliseconds
1	1	0	32 Milliseconds
1	1	1	32 Milliseconds

Table 4-7: Selection values for Debouncer Specification

Auxiliary Digital Input Register *20h*

This read-only register reflects the state of the Auxiliary Digital Inputs to the V535. There is one Auxiliary Digital Input for each axis. Therefore, the 4 axis version of the V535 contains 4 Auxiliary Digital Inputs and the 8-axis version contains 8 Auxiliary Digital Inputs. These digital inputs are in the form of an isolated 10-milliampere current loop. An external input will be read as a **"1"** through this register when it forces the current loop into the ON state.

These digital inputs can be used to generate an interrupt to the host computer. The Auxiliary Digital Input Mask Register is used to enable/disable the individual sources of interrupt generation. The Auxiliary Digital Input Mask Register is used in combination with the Interrupt Mask Register in A16 address space to control the actual generation of the VXI interrupt.

The following diagram shows the bit pattern for the Auxiliary Digital Input Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read-Only	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	AUX IN 8	AUX IN 7	AUX IN 6	AUX IN 5	AUX IN 4	AUX IN 3	AUX IN 2	AUX IN 1

For **D16** transfers, Bits 15-0 are accessed at **22h**.

Auxiliary Digital Input 8 - 1 Bits 7-0

These read-only bits reflect the state of the 8 Auxiliary Digital Inputs. There is one digital input per axis. For the 4 axis V535, only digital inputs 1 through 4 are used. For the 8 axis V535, digital inputs 8 through 1 are used. Any bit read back as a "1" indicates that the corresponding current loop is on the ON state.

Auxiliary Digital Input Mask Register 24h

This write/read register is used to control which, if any, of the Auxiliary Digital Inputs may be allowed to generate a VXI interrupt. The V535 provides one Auxiliary Digital Input for each axis. Therefore, the 4 axis V535 contains 4 Auxiliary Digital Inputs and the 8-axis version contains 8.

The assertion of any of the Auxiliary Digital Inputs can be configured to generate a VXI interrupt. Any input mask bit location is this register set to a "1" allows the corresponding digital input to generate an interrupt. A "0" in any input mask bit location disables that digital input from participating in interrupt generation. Once a mask enabled Digital Input is seen by the V535, it appears as the Masked Source bits in this register. After an interrupt is generated, or if polling is used, the interrupt source bits can be cleared to "0" by writing to its corresponding masked source bit location in the registers. Clearing these bits removes the interrupt source and arms the V535 for the next edge of the digital input.

A second level of interrupt enabling must also occur before a VXI interrupt may be asserted as a result of a digital input assertion. The Interrupt Control Register located in A16 address space must also be properly enabled. Please refer to the *"Interrupt Control Register"* section of this manual for additional details.

The following diagram shows the bit layout for the Auxiliary Digital Input Mask Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Write/Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSKD SRC 8	MSKD SRC 7	MSKD SRC 6	MSKD SRC 5	MSKD SRC 4	MSKD SRC 3	MSKD SRC 2	MSKD SRC 1	INPUT MASK 8	INPUT MASK 7	INPUT MASK 6	INPUT MASK 5	INPUT MASK 4	INPUT MASK 3	INPUT MASK 2	INPUT MASK 1

For **D16** transfers, Bits 15-0 are accessed at **26h**.

Masked Source 8 - 1 Bits 15-8

These bits are used for two purposes depending on the direction of the data transfer. A read of these bits reflects any mask enabled Digital Inputs that have been asserted since the corresponding mask bit was set. These sources can then be used to generate an interrupt. A write to these bit locations with the data set to "1" causes the corresponding masked Digital Input interrupt source to be cleared and arms the V535 for subsequent digital input interrupt sources. Note that an 8 axis V535 uses all eight bits and that a 4 axis V535 uses only bits 11 through 8 of this field.

Auxiliary Digital Input Mask 8 - 1 Bits 7-0

These write/read bits are used to enable/disable the generation of a VXI interrupt when the corresponding digital input is asserted. Any bit location set to a **"1"** enables the digital input to generate an interrupt and a **"0"** disable the input from participating as an interrupt source.

Auxiliary Digital Output Register 28h

This write/read register is used to configure and control the 8 Auxiliary Digital Outputs on the V535. There is one Auxiliary Digital Output for each axis. Therefore, the 4 axis V535 contains 4 Auxiliary Digital Outputs and the 8 axis V535 contains 8 Auxiliary Digital Outputs.

The Auxiliary Digital Outputs are isolated photovoltaic (isolated MOSFET) relays. Each relay is capable of sustaining a continuous 200 volt AC or DC open-circuit voltage as well as 500 volts DC, line to ground. The output can drive a load current up to 100 milliamperes and exhibits an ON resistance of 27 ohms, maximum, at 50 milliampere load. The OFF leakage current is 1 microampere or less. The maximum turn-on time for the relay is 2 milliseconds and the maximum turn-off time is 0.5 milliseconds (assuming a 50 milliampere load). These relays are located in sockets in the V535 that facilitates replacement in the case of component damage. Please refer to the *Auxiliary Digital Outputs* section of this manual for additional information on component replacement. The outputs can be activated by one of four sources. These sources include:

Motion Started – The axis pulse train has started

Motion Complete – The axis pulse train has ceased

Number of Pulses – A preselected number of output pulses has occurred on the corresponding axis since the beginning of the pulse train. A 24-bit counter for each axis is provided yielding a selection from 1 to 16,777,215 counts. The values for these counters are loaded into the Pulse Counter Register for each individual axis.

Software Command – The output is activated by software command from the host computer.

The Auxiliary Digital Output Register contains the means by which each axes output is configured. A two-bit field for each output configures its activation. Once an Auxiliary Digital Output is activated, is must be cleared under software control of the host computer. Eight additional bits in this register are used to provide the clearing mechanism.

The following diagram shows the bit pattern for the Auxiliary Digital Output Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Write/Read	CLR 8	CLR 7	CLR 6	CLR 5	CLR 4	CLR 3	CLR 2	CLR 1	OUT 8	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OUT 8 SEL B	OUT 8 SEL A	OUT 7 SEL B	OUT 7 SEL A	OUT 6 SEL B	OUT 6 SEL A	OUT 5 SEL B	OUT 5 SEL A	OUT 4 SEL B	OUT 4 SEL A	OUT 3 SEL B	OUT 3 SEL A	OUT 2 SEL B	OUT 2 SEL A	OUT 1 SEL B	OUT 1 SEL A

For **D16** transfers, Bits 31-16 are accessed at **28h** and Bits 15-0 are accessed at **2Ah**.

Clear 8 - 1 ***Bits 31-24***

These write-only bits are used to clear the Auxiliary Digital Outputs once they are set. Writing a "1" to a bit position causes the corresponding Auxiliary Digital Output to be cleared. Any of these bits set to a "0" when writing to these bits has no effect on the output. These bits are not latched and always read as "0".

Out 8 - 1 ***Bits 23-16***

These write/read bits are used to set and monitor the state of the Auxiliary Digital Outputs. Any of these bits that are read back as "1" indicate that the corresponding Auxiliary Digital Output is asserted. Writing these bits to a "1" will turn on the corresponding Auxiliary Digital Output when it is configured to be activated by software command. Bits 15 through 0 of this register are used to control the activation mechanisms for the outputs. A write operation to these bit locations with the data set to "0" has no effect on the outputs.

Outx Select B/Outx Select A 8 – 1 ***Bits 15 –0***

These sixteen write/read bits are used to control the activation mechanism of the Auxiliary Digital Output for each axis. There are two bits for each axis that determine its activation mechanism. Each axis can be independently configured to suit the particular application. The binary combination of the OUTx SELECT B and OUTx SELECT A controls the configuration for each axes Auxiliary Digital Output as shown in the following table.

Outx Select B	Outx Select A	Output Activation Method
0	0	Software Control Only
0	1	Axis Motion Started
1	0	Axis MotionCompleted
1	1	Axis Pulse Counter Exhausted

Table 4-8: Selection values for Auxiliary Digital Output Activation Specification

Axis Pulse Counter Register *30h (Axis 1)*

The V535 contains eight Axis Pulse Count Registers that are used to control the assertion of the Auxiliary Digital Outputs when a predetermined number of pulses are applied on a particular axis. The V535 with 4 axis only contains 4 of these registers and the 8-axis version contains 8.

This 24-bit write/read register is used to specify the pulse count before the Auxiliary Digital Output is asserted. This allows for a specification for each axis in the range of 1 through 16,777,215. Once an Auxiliary Digital Output is enabled to assert the corresponding output, the counter starts counting down the value in the pulse counter. The counter is decremented once for each pulse assertion on the individual axis. Once the counter is decremented to zero, the output is asserted. The output remains asserted until it is cleared by writing to the corresponding CLEARx bit in the Auxiliary Digital Output Register.

When the outputs are configured for the Pulse/Direction pair, the Pulse signal is used to increment the counter. When the CW/CCW pair is used, the CW signal is used to increment the counter.

The following diagram shows the bit layout for the Axis Pulse Counter Register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Write/Read	0	0	0	0	0	0	0	0	PCNT 23	PCNT 22	PCNT 21	PCNT 20	PCNT 19	PCNT 18	PCNT 17	PCNT 16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PCNT 15	PCNT 14	PCNT 13	PCNT 12	PCNT 11	PCNT 10	PCNT 9	PCNT 8	PCNT 7	PCNT 6	PCNT 5	PCNT 4	PCNT 3	PCNT 2	PCNT 1	PCNT 0

For **D16** transfers, Bits 31-16 are accessed at **30h** and Bits 15-0 are accessed at **32h**.

Pulse Counter 23-0 ***Bits 23 –0***

These write/read bits are used to determine the number of pulses that are to be output on a given axis before the Auxiliary Digital Output is asserted. This counter must be enabled through the Auxiliary Digital Output Register for the selected channel. Once the counter decrements to zero, the output is asserted, and remains asserted until specifically cleared in the Auxiliary Digital Output Register.

Axis Pulse Counter Register Address Map

The following table indicates the **A24** offset addresses for the eight Axis Pulse Counter registers. (Only Axis 1-4 are used with the 4-channel versions of this module.). The values in parenthesis represent the offset addresses (upper word, lower word) for **D16** transfers.

Axis Number	Axis Pulse Counter Register Address
1	30h (30h, 32h)
2	34h (34h, 36h)
3	38h (38h, 3Ah)
4	3Ch (3Ch, 3Eh)
5	40h (40h, 42h)
6	44h (44h, 46h)
7	48h (48h, 4Ah)
8	4Ch (4Ch, 4Eh)

Table 4-9: Axis Pulse Counter Register address map

Auxiliary Digital Outputs

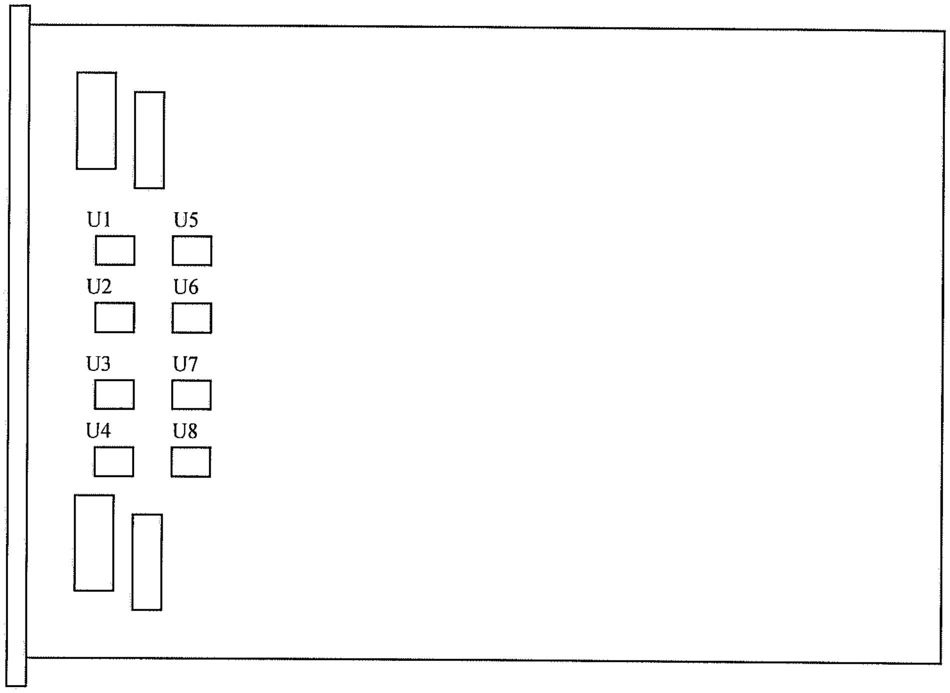
As previously described, the V535 contains an Auxiliary Digital Output per axis that can be activated by one of four sources. These outputs take the form of isolated photovoltaic (isolated MOSFET) relays. Each relay is capable of sustaining a continuous 200 volt AC or DC open-circuit voltage as well as 500 volts DC, line to ground. The output can drive a load current up to 100 milliamperes and exhibits an ON resistance of 27 ohms, maximum, at 50 milliampere load. The OFF leakage current is 1 microampere or less. The maximum turn-on time for the relay is 2 milliseconds and the maximum turn-off time is 0.5 milliseconds (assuming a 50 milliampere load).

The relay for each axis is an International Rectifier PVT412 device. The relays are mounted in flush mount sockets for easy replacement in case of an isolation failure. The following chart and diagram shows the location for each axis' device.

Axis Number	Chip Location
Axis 1	U5
Axis 2	U6
Axis 3	U1
Axis 4	U2
Axis 5	U3
Axis 6	U8
Axis 7	U7
Axis 8	U4

Table 4-10: V535 Axis Device and Chip Locations

Figure 4-1: V535 Isolated Relay Locations

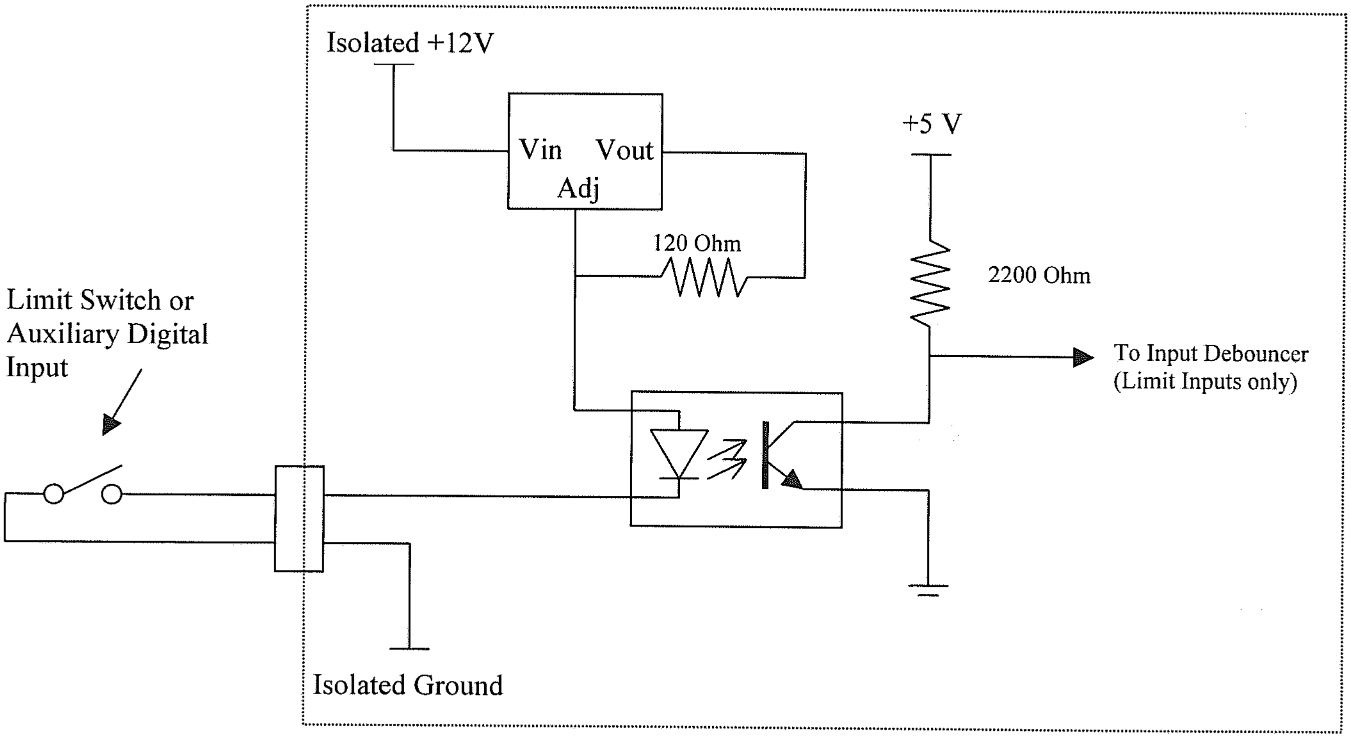


Limit and Auxiliary Digital Inputs

The Limit and the Auxiliary Digital Inputs to the V535 take the form of isolated 10 milliampere current loops. The current loop is energized (turned-on) by causing a completion of the circuit from the digital input to the isolated return for each input signal.

The following diagram shows the input circuit for the Limit and Auxiliary Digital Inputs to the V535.

Figure 4-2: V535 Input Circuit for Limit and Auxiliary Digital Inputs

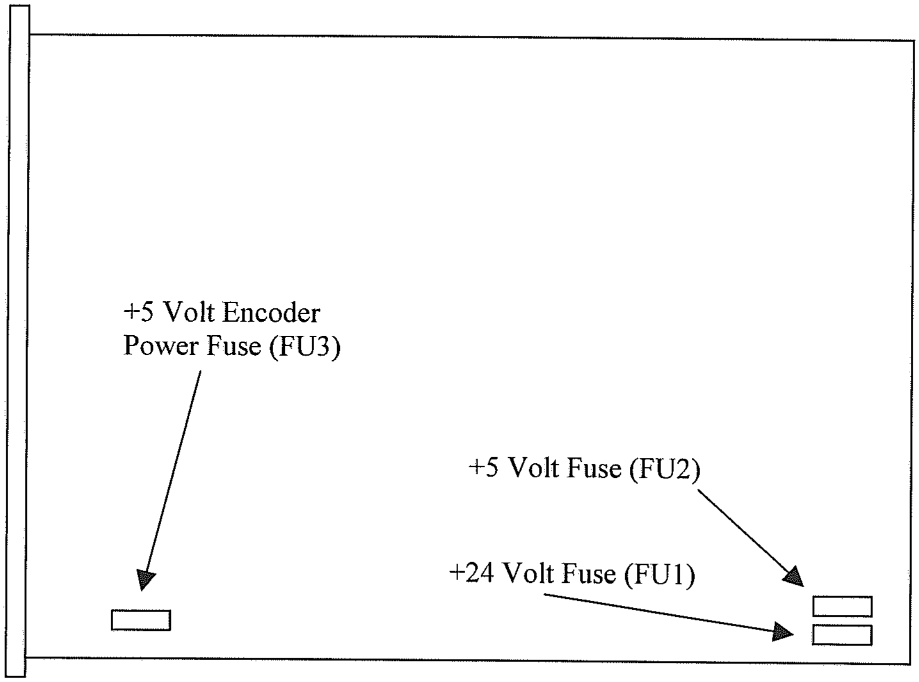


Fuse Locations

The V535 contains 3 fuses for the voltage supplies. Two of these fuses are for the voltage rails supplied by VXI and the third is a fuse for the +5 volt front panel supply. The two voltages received from the VXI backplane are +24 volts and +5 volts. These two fuses are located close to the P2 (lower connector) connector near the rear of the board. The remaining fuse is located toward the front of the board. This is the fuse for the +5 volt connections to the 36-position encoder connectors. This supply can be used to power encoders, but a maximum of 500 milliamps should be drawn from this source. The +5 volt supply on the encoder connector is the same +5 volt supply that powers internal logic on the V535. Exercise extreme caution when making connection to this supply as not to damage the inner circuitry on the card.

The following diagram shows the location of the 3 fuses on the V535.

Figure 4-3: V535 Fuse Locations



Trajectory Profile Generation

The V535 incorporates a chipset manufactured by Performance Motion Devices. This chipset is responsible for executing trajectory profiles based on parameters loaded by the host computer. These commands are interpreted by the chipset and execute the desired profile.

The trajectory profile generator performs the calculations to determine target position, velocity, and acceleration at each phase of the operation. These calculations are performed based on the parameters set up by the host as well as the current profile mode selected. The V535 provides trajectory-profiling modes:

- S-curve Point to Point
- Trapezoidal Point to Point
- Velocity Contouring

The commands used to select the individual profiling modes are as follows:

SET_PRFL_S_CRV selects S-curve profile mode

SET_PRFL_TRAP selects trapezoidal profile mode

SET_PRFL_VEL selects the velocity contouring profile mode

The profile commands are used to select the mode in which each axis is to run. Each axis is independently programmed allowing for multiple profile generation. Generally, the axis should be at reset when altering the profile mode. However, under certain circumstances, the profiling modes can be changes without the axis being at rest. Please refer to the specific profile command descriptions for additional information.

S-Curve Point to Point

The following diagram shows a typical profile for an S-curve motion.

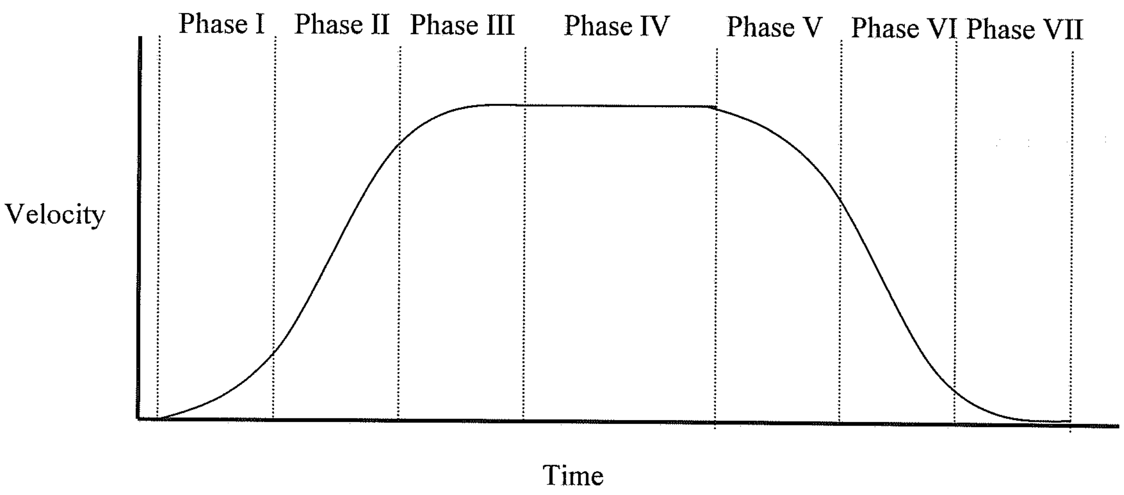


Figure 4-4: V535 S-curve Motion

The S-curve profile drives the axis at the specified jerk until the maximum acceleration is reached and is referred to in the drawing as Phase I. The profiler then drives the axis at a constant acceleration (jerk = 0) through Phase II. Then, the negative value of the jerk specification is used to drive the axis through Phase III such that the axis reaches the specified maximum velocity with acceleration equal to 0. This completes the acceleration phases (Phase I through Phase III). After these three phases are complete, the velocity is constant and the acceleration is zero. At the appropriate time, the profile will then decelerate symmetrically to the acceleration phase such that it arrives at the destination position with acceleration and velocity equal to zero.

There are several conditions where the actual velocity graph of an S-curve profile motion will not contain all of the segments shown in the previous diagram. For example, if the maximum acceleration is not reached before the half-way point to the maximum velocity, then the actual velocity profile will not contain a Phase II of Phase VI segment. A profile of this type is shown below.

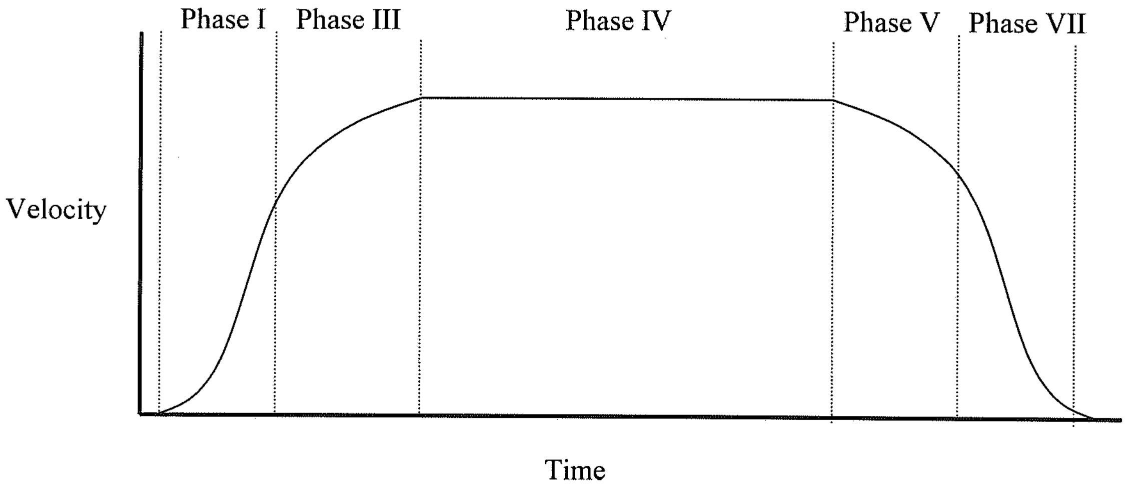


Figure 4-5: V535 Velocity Profile

Another condition that will cause deviation from the 7 phase S-curve profile is one in which a destination position specification that results in the maximum velocity not being reached. In this case, there will be no Phase IV, and there also may be no Phase II and Phase VI, depending on where the profile is terminated.

When an S-curve profile is in motion, the user is not allowed to change ANY of the profile parameters. The axis must be at rest before any parameters are altered. If parameters are changed during motion, a command error occurs any all newly specified parameters are ignored except the position. When the axis is in the S-curve profile mode, the SET_MAX_ACC command should be used to load the maximum acceleration value. The SET_ACC command can not be used for this mode.

An S-curve profile requires 4 parameters from the host for determining the characteristics of the profile. These parameters are summarized in the following table.

Profile parameter	Representation and Range	Units
Destination Position	Signed 32-bit number -1,073,741,824 to 1,073,741,823	Steps
Maximum Velocity	Unsigned 32-bit number* ($1/2^{16}$ scaling) 0 to 1,073,741,823	Steps/Cycle
Maximum Acceleration	Unsigned 16-bit number** ($1/2^{16}$ scaling) 0 to 32,767	Steps/Cycle^2
Jerk	Unsigned 32-bit number*** ($1/2^{32}$ scaling) 0 to 2,147,483,647	Steps/Cycle^3

Table 4-11: V535 S-curve Profile Parameters

*uses $1/2^{16}$ scaling. The command expects a 32-bit number that has been scaled by a factor of 65536 from units of steps/cycle. For example, to specify a velocity of 2.75 steps/cycle, the 2.75 is multiplied by 65536 and the result is the value for the command as a 32 bit number (180,224 or 2C000h).

** uses $1/2^{16}$ scaling. The command expects a 16-bit number that has been scaled by a factor of 65536 from units of steps/cycle^2. For example, to specify an acceleration of .175 steps/cycle^2, the .175 is multiplied by 65536 and the result is the value for the command as a 32 bit number (11,469 or 2CCDh).

*** uses $1/2^{32}$ scaling. The command expects a 32-bit number that has been scaled by a factor of 4,294,967,296 (2^{32}) from units of steps/cycle^3. For example, to specify a jerk of .0075 steps/cycle^3, the .0075 is multiplied by 4,294,967,296 and the result is the value for the command as a 32 bit number (32,212,256 or 1EB8520h).

Trapezoidal Point to Point

The following diagram shows a simple trapezoidal mode profile motion.

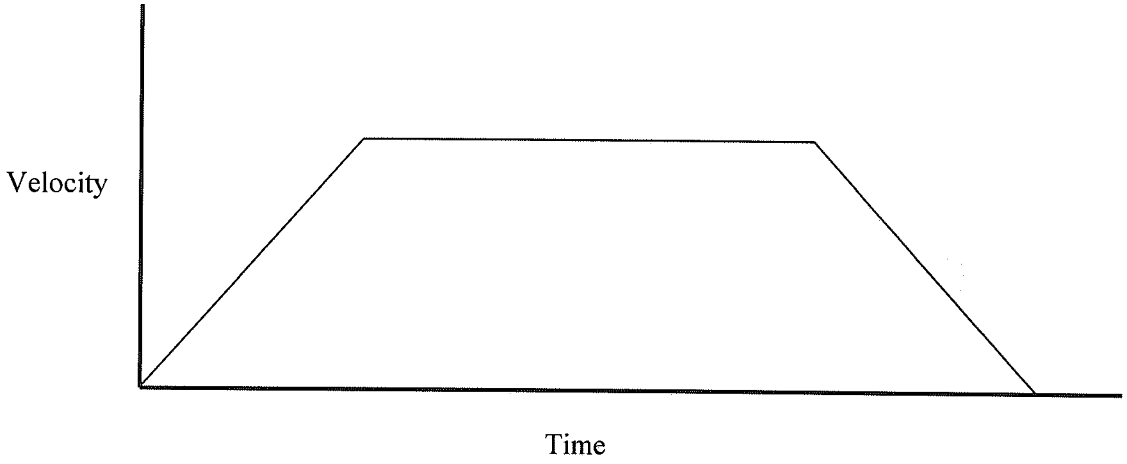


Figure 4-6: V535 Trapezoidal Mode Profile Motion

In the trapezoidal point to point profile mode, the host specifies a destination position, a maximum velocity, a starting velocity, and an acceleration. The trajectory is executed by accelerating at the specified acceleration, beginning at the specified starting velocity, to the specified maximum velocity where it coasts until decelerating such that the destination position is reached with the axis at rest (velocity equal 0). If, due to parametric specification, it not possible to reach the maximum velocity (because deceleration must begin) then the profile will have no coasting phase. In this mode, the acceleration and deceleration rates are the same.

A new maximum velocity and destination position may be specified while the axis is in motion. When this occurs, the axis will accelerate or decelerate toward the new destination position while attempting to satisfy the new maximum velocity position.

Before switching in to the trapezoidal profile mode, make sure that the axis is at rest. When in this mode, the axis must come to a complete stop before setting a new acceleration value. If the acceleration value is changed during a trapezoidal move, a command error is generated and all updated parameters are ignored except the destination position. Also, the starting velocity can not be altered while the axis is in motion.

The following diagram shows a more complex trapezoidal move operation that has the velocity and destination positions altered during the move.

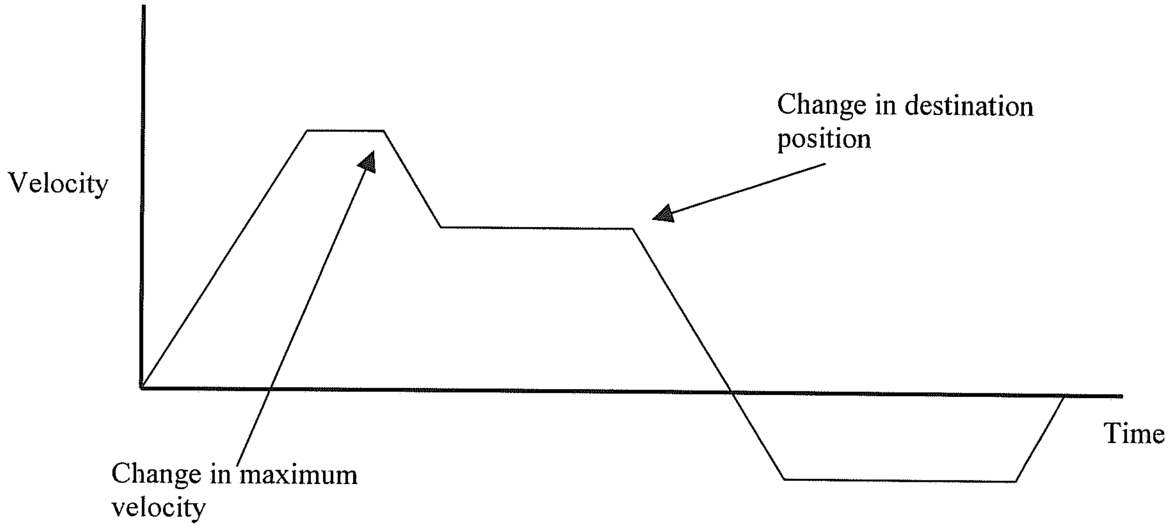


Figure 4-7: V535 Trapezoidal Operation with Velocity and Destination Positions Altered

The next diagram shows trapezoidal profile with a non-zero starting velocity. During the profile generation, an update to the maximum velocity and destination position is made.

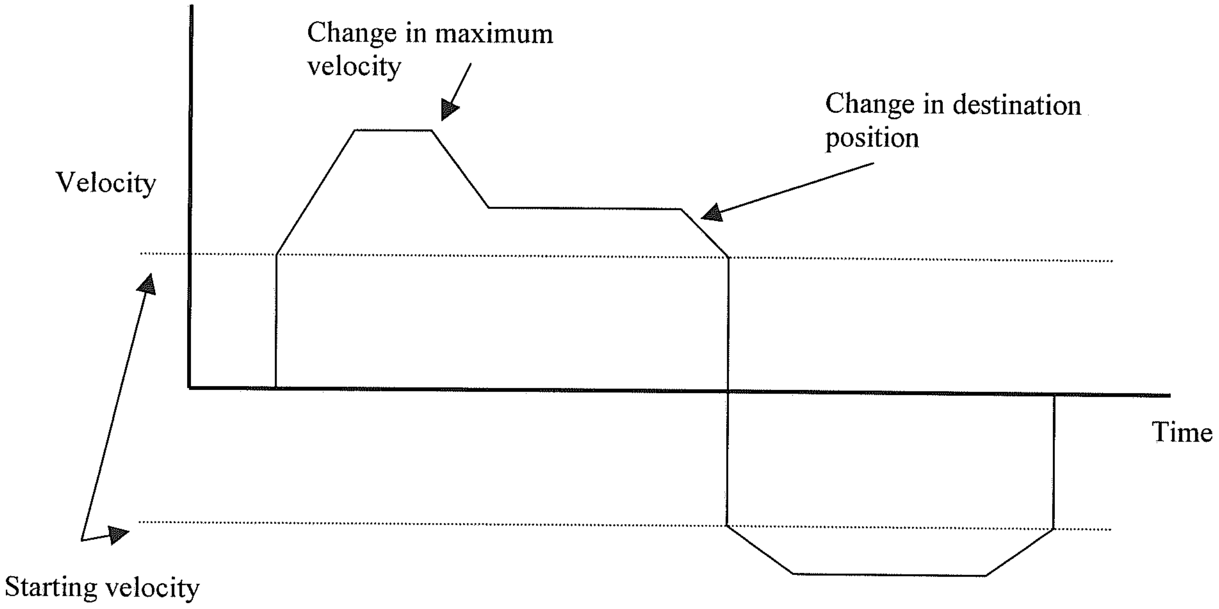


Figure 4-8: V535 Trapezoidal Profile with Non-Zero Velocity

A trapezoidal profile requires 4 parameters from the host for determining the characteristics of the profile. These parameters are summarized in the following table.

Profile parameter	Representation and Range	Units
Destination Position	Signed 32-bit number -1,073,741,824 to 1,073,741,823	Steps
Maximum Velocity	Unsigned 32-bit number (1/2^{16} scaling) 0 to 1,073,741,823	Steps/Cycle
Starting Velocity	Unsigned 32 bit number (1/2^{16} scaling) 0 to 1,073,741,823	Steps/Cycle
Acceleration	Unsigned 32 bit number (1/2^{16} scaling) 0 to 1,073,741,823	Steps/Cycle^2

Table 4-12: V535 Trapezoidal Profile Parameters

Velocity Contouring

During motion in the velocity-contouring mode, the host specifies two basic parameters. These two parameters are the acceleration and the maximum velocity. The trajectory is executed by continuously accelerating the axis at the specified rate until the maximum velocity is reached, or until a new acceleration command is given. The maximum velocity value specified must always be a positive value. Positive acceleration values result in positive motion and negative values result in negative motion. There are no restrictions to changing the profile parameters while the axis is in motion. Note that in this mode, the motion is not bounded by any position specification. It is the responsibility of the host to ensure that acceleration and maximum velocity values result in safe motion of the motor.

The following diagram shows a typical velocity contour profile using this mode.

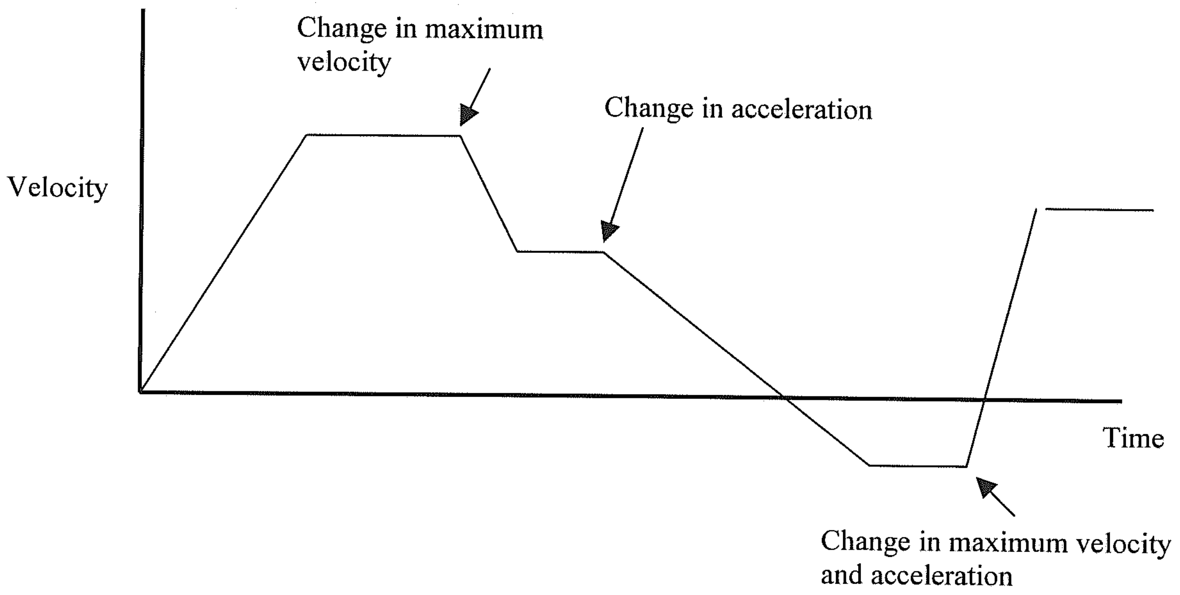

Change in maximum velocity
Change in acceleration
Velocity
Time
Change in maximum velocity and acceleration

Figure 4-9: V535 Typical Velocity Contour Profile

A velocity contour profile requires 3 parameters from the host for determining the characteristics of the profile. These parameters are summarized in the following table.

Profile parameter	Representation and Range	Units
Maximum Velocity	Unsigned 32-bit number (1/2^{16} scaling) 0 to 1,073,741,823	Steps/Cycle
Starting Velocity	Unsigned 32 bit number (1/2^{16} scaling) 0 to 1,073,741,823	Steps/Cycle
Acceleration	Unsigned 32 bit number* (1/2^{16} scaling) -1,073,741,824 to 1,073,741,823	Steps/Cycle^2

Table 4-13: V535 Velocity Contour Profile Parameters

* negative numbers using1/2^{16} scaling are handled the same as positive numbers. For example, to specify an acceleration value of –1.95 steps/cycle^2, -1.95 is multiplied by 65536 and the result is specified with the command (-127,795 or FFFE0CCDh).

Trajectory Control

Halting the Trajectory Profiler

Normally, each of the aforementioned trajectory profiles is executed with the user-supplied parameters until the requested operation is complete. For example, a trapezoidal profile is executed until the destination position is reached and the axis has a velocity equal to zero. In some cases it may be desired to halt the trajectory manually. This may be due to safety reasons, or simply to achieve a desired profile. This can be done through the V535 in one of two ways; abrupt stop or smooth stop.

Abrupt stops are accomplished by executing the STOP command. This command immediately stops the trajectory profile generator by setting the velocity of the axis to zero. This will cause the axis to abruptly stop and is typically used in an emergency situation when no deceleration of the axis is desired.

Smooth stops are accomplished by executing the SMOOTH_STOP command. This command causes the trajectory profile generator to decelerate at the specified acceleration rate until an axis velocity of zero is reached. The deceleration profile is symmetric to the acceleration phase of the profile. If the profile mode selected was S-curve and a SMOOTH_STOP command was given, the profile will decelerate in a manner exactly equal and opposite of the acceleration phase.

The STOP command is available in all profiling modes, but the SMOOTH_STOP is available in all profiling modes except electronic gear.

Care should be taken when using the STOP command due to the large and abrupt nature in the change of motion.

Motion Status Indication

Several mechanisms exist to indicate the status of each axis. Either through polling or interrupt one can determine the motion status of each axis. Two status bits are provided in the Stepper Motor Interface Controller (SMIC) status word to indicate motion status. These two bits are called Motion Complete and In-Motion.

The Motion Complete bit is controlled interactively by the SMIC and the host. After a commanded motion operation completes, the SMIC sets the Motion Complete status bit. The host can then poll for this bit to determine the completion of a motion operation, or if desired, enable the V535 to generate an interrupt when complete. In either case, once the host has recognized a motion complete, the host must clear the Motion Complete bit, enabling the bit to indicate the end of motion for subsequent move operations.

The following is a list of conditions that causes the Motion Complete bit to be set:

- Profile has reached the destination position (point-to-point modes only)
- Axis trajectory reaches a velocity of zero and the current velocity command is zero
- SMOOTH_STOP command is executed and the axis trajectory reaches a velocity of zero
- STOP command is executed
- Limit Switch condition occurs

The In-Motion bit is similar to the Motion Complete bit except that it continually indicates the status of the axis without interaction with the host. Also, this bit is used exclusively for polled operations and it cannot be used to generate an interrupt to the host.

Note that the Motion Complete and In-Motion status bits indicate the state of the trajectory generator, not the actual motor. Even if the trajectory generator has completed a motion, the actual axis position may or may not be at rest depending on motor stability and other system conditions.

Please refer to the *Axis Status* section of this manual for a full description of the axis status word.

Parameter Loading and Updating

Various profile and motor control parameters must be specified by the host for an axis to be controlled in the desired fashion. To facilitate precisely controlled synchronized operation, these parameters and related control commands are loaded into the Stepper Motor Interface Controller (SMIC) using a double-buffered scheme. With this mechanism, the parameters and control commands specified are loaded into the first stage of the double buffer. The second stage of the double buffer is the active register. Data is move from the first stage to the second stage when an update signal is received.

The update signal can consist of either a manual update command or one of several conditional breakpoints. Regardless of the update mechanism utilized, the occurrence of the update signal causes data located in the first stage of the double buffer to be moved into the second stage, the active registers. Before an update signal occurs, loading the first stage of the double buffer register or executing the double buffered commands will have no effect on the system behavior.

The registers inside the SMIC that are double buffered are listed below.

Register Name	Command to Set Value
Destination Position	SET_POS
Maximum Velocity	SET_VEL
Acceleration	SET_ACC
Maximum Acceleration	SET_MAX_ACC
Jerk	SET_JERK

Table 4-14: V535 Double Buffered SMIC Registers

Manual Update

There are two methods of manually updating the double buffered parameters and commands. One method is used for a single axis update and the other for a multiple axis update. The updates through the Stepper Motor Interface Controller (SMIC) can update only those axes that are in its specific group. To execute an update command that controls all 8 axis, one must use the Multi-Axis Command Register.

The single axis immediate update, which is executed using the UPDATE command, forces the parameters for the current axis to be updated at the next cycle time. The multiple axis immediate update, which is specified using the MULTI_UPDATE command, causes multiple axes to be updated simultaneously. This can be a useful technique when synchronized multi-axis profiling is required. The MULTI_UPDATE command consists of a bit mask with a bit assigned to each axis. Executing this command has the same result as switching to each individual axis and executing the UPDATE command.

Breakpoint Update

A breakpoint is a convenient way of programming a profile or other double-buffer parametric updates to occur upon a specific condition. These conditions are referred to as breakpoints. There are two types of breakpoints, those which have a 32-bit comparison value associated with them and those which have an action associated with them. For those breakpoints that require the comparison value, a 32-bit comparison value is loaded into the breakpoint register and then a breakpoint condition is specified. For those breakpoints that are based on an action only, no comparison needs to be specified.

The double-buffered registers and commands are updated when the breakpoint condition is satisfied. The following is a list of available breakpoint conditions.

- Positive Target Position Breakpoint

  A 32 bit position breakpoint is specified which results in the parameters being updated when the current target position, which is the instantaneous desired axis position output from the profile generator, equals or exceeds the specified breakpoint value. This breakpoint is set through the SET_POS_BRK command.

- Negative Target Position Breakpoint

  A 32 bit position breakpoint is specified which results in the parameters being updated when the current target position, which is the instantaneous desired axis position output from the profile generator, equals or is less than the specified breakpoint value. This breakpoint is set through the SET_NEG_BRK command.

- Positive Actual Position Breakpoint

A 32 bit position breakpoint is specified which results in the parameters being updated when the current actual target position, which is the instantaneous position of the actual axis hardware, equals or exceeds the specified breakpoint value. This breakpoint is set through the SET_ACTL_POS_BRK command.

- Negative Actual Position Breakpoint

  A 32 bit position breakpoint is specified which results in the parameters being updated when the current actual target position, which is the instantaneous position of the actual axis hardware, equals or is less than the specified breakpoint value. This breakpoint is set through the SET_ACTL_NEG_BRK command.

- Time Breakpoint

  A 32 bit time breakpoint is specified which results in the parameters being updated when the number of cycles executed since the SMIC was reset (the current SMIC time) is equal to the time breakpoint value. The number of cycles continually increases until it rolls over from (2^{32} –1) back to 0. The time breakpoint is set through the SET_TIME_BRK command.

- Motion Complete Breakpoint

  A breakpoint can be specified which results in the parameters being updated when the previous motion is complete. This is indicated by the setting of the Motion Complete bit in the status word for the desired axis. When using this mode, no comparison data is required. This breakpoint is enabled through the SET_MTN_CMPLT_BRK command.

When one of these conditions has been programmed and satisfied, the double-buffered parameters are transferred into the active registers. A global breakpoint disable command exists to prevent the automatic updating of the active registers when a breakpoint is reached. This is discussed in more detail in the following section on disabling automatic updates.

The above breakpoint modes are particularly useful during multi-axis operations. This facility allows the next pre-loaded profile command, as specified by the host, to be activated at the precise position or time required, with no delay incurred to send an update command.

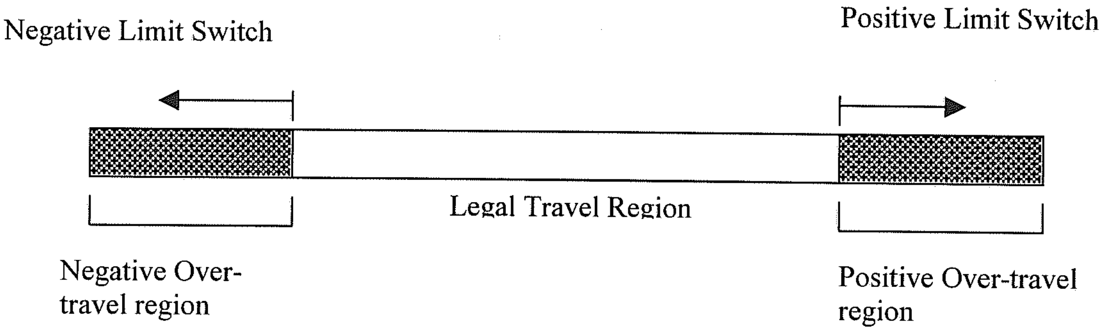
Once a breakpoint condition has been satisfied, it is no longer active. To allow subsequent breakpoint activity, the host must re-enable the breakpoint. After a breakpoint is reached, the first stage of the double-buffer registers remains intact, only the active register contents are altered.

Disabling Automatic Profile Updating

Normally, when a breakpoint condition has been satisfied, it causes the profile and other double-buffer parameters to be automatically updated by moving the data to the active registers. For certain applications it may be desirable to use the breakpoint mechanism to notify the host of a specific condition, but not to have the automatic update feature enabled. To disable the automatic update feature, the SET_AUTO_UPDATE_OFF command is used. When the automatic update feature is disabled, the only way to force register updates is with either the UPDATE command or the MULTI_UPDATE command.

Travel Limit Switches

The Stepper Motor Interface Controller (SMIC) incorporated into the V535 supports motion travel limit switches that can be used to automatically recognize an end of travel condition. The following diagram illustrates an axis with travel limit switches installed. he area within the limit switches is the legal area of motion.



The V535 provides two mechanisms that can be used in conjunction with the over-travel limit switches.

1.) The host can be automatically notified through an interrupt that an axis has entered an over-travel condition. Once the host is notified it can take appropriate action to manage the over-travel condition.

2.) Upon entering an over-travel condition, the trajectory generator will be automatically halted so that the motor does not travel further into the over-travel region.

To recover from an over-travel condition the corresponding status bits in the status word should be reset. Please refer to the section of this manual on *Axis Status* for additional information on resetting this condition. Once this reset is complete, the host can command a trajectory move operation to bring the offending axis out of the over-travel region.

Only one over-travel signal can be processed at a time. For example, if the negative limit switch is activated, the corresponding status bits must be cleared, and the axis moved into the legal travel region before a positive over travel limit switch will be recognized.

Axis Timing

The timing of the stepper motors is controller by circuitry on the V535 referred to as the Stepper Motor Interface Controller (SMIC). Each axis that the SMIC controls receives a period of time in which available computational power is assigned. The amount of time required for the SMIC to perform one complete pass of calculations for all of the axes is known as the cycle time. The cycle time is important because it determines the rate at which trajectory profiles are updated. The cycle time is 327.68 microseconds. This value is typically rounded off to 330 microseconds. All velocities, acceleration, and jerk values are relative to this cycle time through the various trajectory generator modes that actually generate the axis motion.

The following two examples show how to correlate the cycle time of the SMIC to the parametric values specified for profile generation.

Example 1:

To determine the velocity of a given axis in units of steps/second, the conversion ratio of 1 second = 3,030 cycles (3,030 cycles/second = 1 cycle every 330 microseconds) is used. Therefore, if the desired maximum velocity is 12,345 steps/second, convert to units of steps/cycle by dividing by 3,030, yielding a value of 4.07425. Then, the value to send to the SMIC using the SET_VEL command would be 65,536 times this amount since the velocity command uses1/2^{16} scaling. The actual value sent with the SET_VEL command would be 267,010 (41302h).

Example 2:

To determine the acceleration of a given axis in units of steps/second^2, the conversion ratio of 1 second = 3,030 cycles must be used. However, this conversion ratio is for cycles, not cycles^2. One must account for this when performing the calculation. Assume that the desired acceleration to be provided is 67,890 steps/second^2, convert to units of steps/cycle^2 by dividing by 3,030^2 (9,180,900), yielding a value of .00739469. Then the value to send to the SMIC using the SET_ACC (or SET_MAX_ACC) command would be 65,536 times this amount since the acceleration command uses1/2^{16} scaling. The actual value sent with the command is 485 (1E5h).

Host Communication

The V535 incorporates a chipset to facilitate the interface to the stepper indexer. This interface is referred to as the Stepper Motor Interface Controller (SMIC). The SMIC is accessed through an internal 8-bit data path. This data path is hidden from VXI by a structure of registers and interface circuitry. The interface to VXI is simplified by this circuitry and presents a cleaner programmable interface to the host.

There are three basic types of transfers between VXI and the SMIC. These transfers fall into the categories of Command Write, Data Write and Data Read. A Command Write operation involves the transfer of a single byte of data to the SMIC. The data transferred from this type of operation is defined in the *Stepper Motor Indexer Command Reference* section of this manual and specified the operation that the SMIC is to execute. The Data Write operation involves the transfer of two bytes of data to the SMIC. These two bytes are transferred as most significant byte first and the least significant second. As described in the Command Reference section of this manual, some commanded operations require two data bytes to define a parameter and some require four. The Data Read operation involves the transfer of two data bytes from the SMIC. These two bytes are transferred as most significant byte first and least significant second. As described in the Command Reference section of this manual, some commanded operations return two data bytes and some return four.

All of these byte transfers are hidden from the user by the VXI interface. The VXI interface presents data in 16 and 32-bit formats. The hardware on the V535 is responsible for moving the data from the VXI registers to/from the 8-bit data path for the SMIC.

Checksum Patterns

The communication path from the VXI interface to the SMIC can have a checksum pattern enabled to validate any commands and data sent to/from the SMIC. There are enable bits in each Command/Status Register for each group to enable or disable this function. The checksum can be used to verify that the expected command and data were received correctly by the SMIC.

This checksum consists of the 16-bit sum of all previous communications that have occurred for the associated command. The command byte, from the Command Write operation, is included in the checksum calculation. Data words are added to the checksum value as they are transferred.

For example, assume a SET_VEL (set velocity) command, which takes two 16-bit data words, was sent with a data value of FEDCBA98h, the checksum would be as follows:

0011	(code for SET_VEL command)
+ FEDC	(high data word)
+ BA98	(low data word)

1B985	checksum = B985 since only the lower 16-bits are used

Reading the checksum is an option. Recovering from a checksum error depends on the nature of the command. Read and Write operations can always be re-transmitted, while a command resulting in an action may or may not be retried, depending on the command and the state of the axis.

Illegal Commands

If the Stepper Motor Interface Controller (SMIC) receives a command that is illegal, it will signal this event by generating a checksum of zero, regardless of the illegal command value or the value of any subsequent data returned as part of the illegal command sequence. With the checksum enabled, the host can determine when an illegal sequence has occurred.

Command Errors

If a command or command sequence is sent to the SMIC that is not valid at the time it is executed, but is valid at other times, this command generates a command error. When a command error occurs, this condition is indicated by the Command Error bit of the axis status word. Please refer to the *Axis Status* section of this manual for additional information.

The following is a list of the command sequences that can generate a Command Error.

1.) Changing and updating the acceleration (SET_ACC and UPDATE commands) when in the trapezoidal profile mode and the axis is still in motion.

2.) Changing and updating either the velocity, maximum velocity, or jerk (SET_VEL, or SET_MAX_ACC, or SET_JERK command followed by UPDATE command) when in the S-curve profile mode while the trajectory is in motion.

3.) Commanding a move operation in the same direction as a limit switch condition during a trapezoidal or S-curve mode operation. For example, if a traveling in a positive direction caused a limit switch violation, a move further in that direction would be ignored and a command error generated.

Once a command error has been set, all illegal profile changes are ignored. If additional parameters are also changed such as position, or any other values as part of the UPDATE command, these parameters are not rejected and become active at the time of the UPDATE command.

Axis Status

The Stepper Motor Controller Interface (SMIC) provides a status word for each axis. The status word is a 16-bit data value that can be obtained by executing the GET_STATUS command. The following is the bit layout for this status word.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	X	X	Current Axis Hi	Current Axis Lo	X	In Motion	Axis Active	Motor Active	Command Error	Neg Limit	Pos Limit	Motion Error	Pos Cap Rcvd	Updt Brk Rcvd	Pos Wrap	Motion Comp

Current Axis Hi *Bit 13*

Current Axis Hi is used in conjunction with Current Axis Lo to determine the current axis being accessed. The binary combination of these bits reflects the actual axis number.

Current Axis Lo *Bit 12*

Current Axis Lo used in conjunction with Current Axis Hi determines the current axis being accessed. The binary combination of these bits reflects the actual axis number.

In Motion *Bit 10*

This bit is returned as a one "1" when the current axis is in motion. If the axis is at rest, this bit is returned as a "0".

Axis Active *Bit 9*

The Axis Active bit is returned as a "1" when the current axis is ON and a "0" when the current axis is OFF.

Motor Active *Bit 8*

Motor Active is returned as a "1" when the motor is ON and a "0" when the motor is OFF.

Command Error *Bit 7*

Command Error is returned as a "1" when a host communication sequence causes an error in the command. A "0" in this location indicates no command error.

Negative Limit Switch *Bit 6*

Negative Limit Switch is returned as a "1" when a negative (CCW) over-travel limit switch is activated. This bit is returned as a "0" if no over-travel limit exists.

Positive Limit Switch *Bit 5*

Positive Limit Switch is returned as a "1" when a positive (CW) over-travel limit switch is activated. This bit is returned as a "0" if no over-travel limit exists.

Motion Error *Bit 4*

Motion Error is returned as a "1" when the maximum position error set for a particular axis has been exceeded. A value of "0" is returned for this bit location if no error exists.

Position Capture Received *Bit 3*

Position Capture Received is returned as a "1" when the encoder index pulse has been captured. If no position capture has occurred, this bit is returned as a “0”

Update Breakpoint Reached *Bit 2*

Update Breakpoint Reached is returned as a”1” when a breakpoint condition has been satisfied. A value if “0” if returned for this bit is the breakpoint has not yet been reached.

Position Wrap *Bit 1*

Position Wrap is returned as a “1” when the axis position wraps. A “0” value returned for this bit location indicates no wrap has occurred. This error is generated specifically when travelling in a positive direction past the position 1,073,741,823 and the axis position wraps around to –1,073,741,824.

Motion Complete *Bit 0*

Motion Complete is returned as a “1” when an axis profile is complete. A value of “0” is returned if the axis is still in motion.

In the status word, bits 8 through 10 and 12 through 13 indicate current status of their respective conditions and do not need to be reset by the host. Bits 0 through 7 indicate the various status flags that can also be used to generate interrupt sources. These flags are set by the SMIC and must be reset by the host in order to correctly represent subsequent conditions.

Miscellaneous Mode Status Word

There is a second status word available that indicates the current status of various mode settings and conditions. The miscellaneous mode status word is a 16-bit data word that can be obtained by executing the GET_MODE command. The following diagram shows the bit layout for this word.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Phase 2	Phase 1	Phase 0	Profile 1	Profile 0	Auto Update	Pulse Mode	RSVD	Error Stop	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

Phase 2-0 *Bits 15-13*

The Phase 2 through 0 bits reflects the current phase of an S-curve profile move. A phase of 0 indicates that the profile has not yet started. A phase of 1 through 7 indicates the phases corresponding to the description under the *S-curve Profiling* section of this manual.

Profile 1-0 *Bits 12-11*

The Profile1 through 0 bits reflect the current trajectory profile mode. The binary combination of these bits determine the mode as follows:

Profile 1	Profile 0	Trajectory Profile
0	0	Trapezoidal
0	1	Velocity Contouring
1	0	S-Curve

Table 4-15: V535 Trajectory Profile

Auto Update ***Bit 10***

Auto Update is returned as a "1" when the automatic update feature is disabled. It is returned as a "0" when the automatic update feature is enabled.

Pulse Mode ***Bit 9***

Pulse Mode is returned as a "1" when high-speed mode is selected and a "0" when low speed mode is selected.

Error Stop ***Bit 7***

Error Stop is returned as a "1" when stop on motion error is enabled and a"0" when stop on motion error is disabled.

Stepper Motor Interface Controller Interrupt Sources

In many instances, during axis motion or at other times, it is useful to have the Stepper Motor Interface Controller (SMIC) signal the host that a special condition has occurred and host intervention may be required. This is generally a more efficient mechanism than polling the various registers on the SMIC for condition status.

Several SMIC conditions may occur that can result in the generation of an interrupt. Whether these conditions in fact interrupt the host is controlled through individual mask bits for each condition. Please note that the Interrupt Control Register must be set up appropriately before these interrupt sources can generate VXI interrupts. Please refer to the *Interrupt Control Register* section of this manual for additional information.

The interrupt source conditions correspond to bits 7 through 0 of the status word (the axis event flags) described in the previous section. These conditions are summarized below.

- Motion Complete

  This interrupt source occurs when a profile is complete.

- Wrap-Around Condition

  This interrupt source occurs when the axis position overflows the position register and causes it to wrap from full scale positive to full scale negative.

- Breakpoint Reached

  This interrupt source is generated when a breakpoint condition has been satisfied.

- Position Capture Received

  This interrupt source is generated when the encoder index pulse has been successfully captured.

- Motion Error

  This interrupt source is generated when the maximum position error set for a particular axis has been exceeded.

- Negative Limit Switch

  This interrupt source occurs when a negative (CCW) over-travel condition is found.

- Positive Limit Switch

  This interrupt source occurs when a positive (CW) over-travel condition is found.

- Command Error

  This interrupt source is generated when a host communication sequence causes a command error condition.

When one of the interrupt source conditions occur from the SMIC on a given axis, an interrupt is generated to VXI, if enabled. At this point, the host can respond to the interrupt but is not required to do so. When the host has completed servicing the interrupt, it must send a command to the clear the interrupting axis by using the RST_INTRPT command. This command includes a clear bit 'mask' as a passed parameter that allows for clearing of individual interrupt source conditions.

If one is programming or setting up parameters on axis #1 and an interrupt occurs on axis #2, it is the responsibility of the host to change to axis #2 to ascertain information regarding the interrupting axis. If more than one axis generates an interrupt condition at the same time, the axis with the lowest number will generate the interrupt first.

The following is a summary of the commands used to manage interrupts on the SMIC. Please refer to the *Stepper Motor Indexer Command Reference* section of this manual for additional information regarding use of these commands.

- SET_INTRPT_MASK

  This command is used to set the interrupt conditions mask. Some of the interrupt source conditions can be enabled and some disabled.

- GET_INTRPT

  This command is used to return the status of the interrupting axis, which includes the interrupting axis number. The current axis number is not affected by this command.

- SET_I

  This command changes the current axis number to the one that is generating the interrupt source. This command serves the dual purpose of determining which axis is interrupting and automatically switching.

- RST_INTRPT

  This command clears the individual interrupt source condition for the interrupting axis. The current axis number is not affected by this command.

The following represents a typical sequence of interrupt conditions and what actions the host may take to resolve them. For this discussion, assume that an axis has hit a "hard stop" causing an instantaneous motion error, as well as a positive limit violation. Also, assume that the interrupt mask for this axis was setup so that either motion errors or limit switch violations will cause an interrupt.

Event	Action
Motion error and limit switch violation generates an interrupt	Host sends SET_I command
Interrupting axis status returned by SMIC, current axis set to interrupting axis	Host detects motion error and limit switch flags are set, recovers from motion error first. Host sends RST_INTRPT 00EF, clearing the motion error bit
SMIC clears motion error bit and disables interrupt source line	-
Due to limit switch violation still in effect, SMIC generates another interrupt source	Host sends SET_I command
Interrupting axis status returned by SMIC, current axis set to interrupting axis	Host detects that limit violation flag is set, performs recovery for limit violation. Host sends RST_INTRPT 00DF, clearing limit violation bit
SMIC clears limit switch violation and disables interrupt source line	-

Table 4-16: V535 Sequence of Interrupt Conditions

At the end of this sequence all status bits are clear, the interrupt sources are inactive, and there are no interrupts pending.

Note that one can process more than one interrupt at a time. It is acceptable, that once a determination has been made that more than one interrupt source is pending, to clear multiple interrupt sources at once by setting the appropriate mask bits when the RST_INTRPT command is issued. The RST_INTRPT and GET_I commands are only valid there is an interrupt pending. If no interrupt is pending, then commands such as GET_STATUS and RST_STATUS should be used.

Indexer Control Pulse Signal Generation

The V535 provides two signals per axis which determine the axis position at any given moment. By default, these signals are Direction and Pulse. The Direction signal determines which direction the motor is to rotate and the Pulse signal indicates the number of steps to take in the indicated direction. Through software programming, these two signals for indexer control can be changed to Clockwise and Counterclockwise. Each time a pulse is applied on the Clockwise signal the motor takes one step in that direction and each time a pulse is applied on the Counterclockwise signal the motor takes one step in that direction. Programming of the signal output convention is selected through the Digital I/O Control Register.

The remainder of this discussion will focus on the signal pair of Pulse and Direction. Each axis that the V535 controls is driven by these two signals. Strap selections on the module allow these two signals to be either RS-422 differential signaling or open-collector type signaling. Please refer to the *Selecting the Stepper Output Signals* section of this manual for the actual selection process.

The pulse signals output by the V535 consist of a precisely controller series of individual pulses each of which represent a desired increment of motion. The signal is always output as a square wave pulse train with a 50% duty cycle regardless of pulse rate. A step or pulse is considered to have occurred when the pulse signal has made a transition from high to low. For the RS-422 connection, this relates to the + pulse signal being at a more positive potential than the – pulse signal and then making a transition to the – pulse signal being at a more positive potential than the + pulse signal. The Direction signal is synchronized with the Pulse signal at the moment each pulse transition occurs. The Direction signal is encoded such that a high value (voltage level high) indicates movement in the positive direction and a low value (voltage low level) indicates movement in the negative direction.

The V535 supports two separate pulse rate modes, known as the standard speed mode which operates at 48.8Ksteps/second and the high-speed mode which operates at 1.55Msteps/second. For full-step and half step applications, as well as pulse and direction applications that have a maximum velocity of about 48.8Ksteps/second, the standard speed range should be selected.

The standard speed mode of 48.8Ksteps/second is selected by using the SET_OUTPUT_STNDRD command and the SET_OUTPUT_HIGH is used to select the 1.55Mstep/second speed mode. The current output rate settings can be obtained by using the GET_MODE command.

Pulse Generation Control

The rate at which pulses are output by the V535 is usually determined by the particular trajectory profile parameters being requested by the host. In addition to the trajectory profile generator, a separate method exists for enabling and disabling pulse generation. This method is used to turn the motor ON and OFF through use of the MTR_ON and MTR_OFF commands. The MTR_OFF causes the trajectory generator to immediately discontinue further pulse generation until a MTR_ON command is executed. As long as the motor is in the OFF state, any further trajectory commands will have no effect until the motor is turned ON. The current motor status can be obtained through use of the GET_STATUS command and evaluating the returned status word (Bit 8). If the motor it turned ON by the host using the MTR_ON command, the motor will stay at rest until a new trajectory move is loaded and initiated.

Along with the manual control of the motor ON and OFF commands, the Stepper Motor Interface Controller (SMIC) can also turn the motor OFF. This can occur if the SMIC detects a motion error condition when the auto-stop feature is enabled.

Chapter 5: Stepper Motor Indexer Command Reference

This command reference refers to Axes 4 through 1. For a V535 with 8 axis, the same command definitions apply but should be directed to the set of Group B registers.

Axis Control Commands

SET_1	**Set current axis to #1**
Data/direction:	1/read
Encoding:	01h
Axis acted on:	Set by command
Available on:	All axis
Double buffered:	No

SET_1 changes the current axis number to 1. All commands that operate on the 'current axis' will be affected by this command. Once this command is executed to set the current axis, the axis status word is returned. Please refer to the GET_STATUS command for the data word returned for the axis status.

SET_2	**Set current axis to #2**
Data/direction:	1/read
Encoding:	02h
Axis acted on:	Set by command
Available on:	All axis
Double buffered:	No

SET_2 changes the current axis number to 2. All commands that operate on the 'current axis' will be affected by this command. Once this command is executed to set the current axis, the axis status word is returned. Please refer to the GET_STATUS command for the data word returned for the axis status.

SET_3	**Set current axis to #3**
Data/direction:	1/read
Encoding:	03h
Axis acted on:	Set by command
Available on:	All axis
Double buffered:	No

SET_3 changes the current axis number to 3. All commands that operate on the 'current axis' will be affected by this command. Once this command is executed to set the current axis, the axis status word is returned. Please refer to the GET_STATUS command for the data word returned for the axis status.

SET_4	**Set current axis to #4**
Data/direction:	1/read
Encoding:	04h
Axis acted on:	Set by command
Available on:	All axis
Double buffered:	No

SET_4 changes the current axis number to 4. All commands that operate on the 'current axis' will be affected by this command. Once this command is executed to set the current axis, the axis status word is returned. Please refer to the GET_STATUS command for the data word returned for the axis status.

SET_I	**Set current axis to the interrupting axis**
Data/direction:	1/read
Encoding:	08h
Axis acted on:	Interrupting Axis
Available on:	All axis
Double buffered:	No

SET_I changes the current axis to the one that is generating the interrupt request. All commands that operate on the 'current axis' will be affected by this command. Once this command is executed to set the current axis to the interrupting axis, the axis status word is returned. Please refer to the GET_STATUS command for the data word returned for the axis status

Profile Generation Commands

SET_PRFL_S_CRV	**Set profile mode to S-curve point to point**
Data/direction:	None
Encoding:	0Bh
Axis acted on:	Current Axis
Available on:	all axis
Double buffered:	No

SET_PRFL_S_CRV sets the trajectory profile mode to S-curve point to point. With this mode, the destination position (SET_POS command), the maximum velocity (SET_VEL command), the maximum acceleration (SET_MAX_ACC command) and the jerk (SET_JERK command) must also be specified. Once in this mode, the trajectory profile generator drives the axis to the specified destination position at the specified jerk while not exceeding the maximum velocity and the maximum acceleration. The axis stays in the selected mode until another profile mode is explicitly set.

While in this profile mode, no parameters can be changed while the axis is in motion. Before setting the current profile mode to S-curve point to point, make sure that the axis is completely at rest.

SET_PRFL_TRAP **Set profile mode to trapezoidal point to point**

Data/direction:	None
Encoding:	09h
Axis acted on:	Current Axis
Available on:	all axis
Double buffered:	No

SET_PRFL_TRAP sets the trajectory profile mode to trapezoidal point to point. With this mode, the destination position (SET_POS command), the maximum velocity (SET_VEL command), the starting velocity (SET_VEL command) and the acceleration (SET_ACC command) must also be specified. Once in this mode, the trajectory profile generator will drive the axis to the specified destination position at the specified acceleration while not exceeding the specified maximum velocity. When in this mode, the position and velocity parameters may be changed on the fly, but the acceleration and starting velocity may not be changed. The axis stays in the selected mode until another profile mode is explicitly set.

Before setting the current profile mode to trapezoidal point to point, ensure that the axis is completely at rest. Also, make sure to NOT change the acceleration until the axis has come to a complete stop.

SET_PRFL_VEL **Set profile mode to velocity contouring**

Data/direction:	None
Encoding:	0Ah
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	No

SET_PRFL_VEL sets the trajectory profile generator to velocity contouring. With this mode, the acceleration (SET_ACC command), the starting velocity (SET_START_VEL command) and the maximum velocity (SET_VEL command) must also be specified. Once in this mode, the trajectory profile generator drives the axis at the specified acceleration while not exceeding the maximum velocity. When in this mode, the acceleration and maximum velocity may be changed on the fly, but the starting velocity may NOT be changed. The axis stays in the selected mode until another profile mode is explicitly set. There are no limitations on changing the profile mode to velocity contouring while the axis is in motion.

In this mode, there are no user-specified limits on the position. It is the responsibility of the user to specify profile parameters that maintain the axis within safe position limits.

SET_POS Set command position

Data/direction:	2/write
Encoding:	10h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

SET_POS sets the final position used during S-curve and trapezoidal trajectory profile generation modes. The position is specified as a signed 32-bit number with the units of steps. The range is –1,073,741,824 to 1,073,741,823. Once this command is executed, the loaded parameter is not utilized until a parameter update command occurs.

SET_VEL Set command velocity

Data/direction:	2/write
Encoding:	11h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

SET_VEL sets the maximum velocity magnitude used during S-curve, trapezoidal, and velocity contouring profile modes. The velocity is specified as an unsigned 32-bit number with units of steps/cycle. The data word scaling is $1/2^{16}$ and the range is from 0 to 1,073,741,823. Once this command is executed, the loaded parameter is not utilized until a parameter update command occurs.

SET_ACC Set command acceleration

Data/direction:	2/write
Encoding:	12h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

SET_ACC set the command acceleration. When in the trapezoidal point to point mode, the acceleration is specified as an unsigned 32-bit number with units of steps/cycle^2, represented using $1/2^{16}$ scaling. The range for this value is 0 to 1,073,741,823. When in the velocity contouring mode, he acceleration is specified as a signed 32-bit number with units of steps/cycle^2 represented in $1/2^{16}$ format, yielding a range of –1,073,741,824 to +1,073,741,823. Once this command is executed, the loaded parameter is not utilized until a parameter update command occurs. This command is used with trapezoidal point to point or velocity contouring.

SET_MAX_ACC	**Set maximum acceleration**
Data/direction:	1/write
Encoding:	15h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

SET_MAX_ACC set the maximum acceleration for a move operation. The acceleration is specified as an unsigned 16-bit number with the units specified in steps/cycle^2 represented using $1/2^{16}$ scaling. The range is from 0 to +1,073,741,823. Once this command is executed, the loaded parameter is not utilized until a parameter update command occurs.

This command is used when the profile mode is set to S-curve point to point.

SET_JERK	**Set command Jerk**
Data/direction:	2/write
Encoding:	13h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

SET_JERK sets the command jerk used during the S-curve profile generation mode. The jerk parameter is specified as an unsigned 32-bit number with units in steps/cycle^3. The scaling is $1/2^{32}$ and the range is from 0 to 2,147,483,647. Once this command is executed, the loaded parameter is not utilized until a parameter update command occurs.

SET_START_VEL	**Set Starting Velocity**
Data/direction:	2/write
Encoding:	6Ah
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	No

SET_START_VEL sets the minimum velocity allowed. This command is used for trapezoidal and velocity contouring profiling modes and is very useful in systems that may be induced into oscillation due to operating speeds that are too low. The starting velocity is specified as an unsigned 32-bit number with units of steps/cycle. The data word scaling is $1/2^{16}$ and the range is from 0 to 1,073,741,823.

The starting velocity must always be set to be less than the maximum velocity set using the SET_VEL command.

This command is not used for S-curve or electronic gear profile modes.

The starting velocity parameter is not double buffered. This parameter is updated immediately after execution and does not require the UPDATE command.

STOP	**Abruptly stop the current axis motion**
Data/direction:	none
Encoding:	46h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

STOP instructs the current axis to cease motion by setting the target velocity to zero. Once this command is executed, it will not be performed until a parameter update occurs. After the update occurs, the axis trajectory generator will stop and the motion complete bit is set to true. This command is useful for stopping an axis immediately.

SMOOTH_STOP	**Smoothly stop current axis motion**
Data/direction:	none
Encoding:	4Eh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

SMOOTH_STOP stops the current axis by setting the desired velocity to zero which results in a controlled deceleration of the axis to zero. The deceleration profile will mirror the acceleration profile for the current profile mode.

This command does not function when the profile mode is set to Electronic Gear.

SYNCH_PRFL	**Set target position equal to the actual position**
Data/direction:	none
Encoding:	47h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	Yes

SYNCH_PRFL sets the trajectory profile generator target position (in steps) equals to the actual axis position (in encoder counts), which clears the error. This command is available for all profiling modes. . Once this command is executed, the loaded parameter is not utilized until a parameter update command occurs.

The SYNCH_PRFL command does not set the target velocity to zero. If the axis should not move after a SYNCH_PRFL command, then a STOP command should also be executed in addition to the

SYNCH_PRFL command. This command is only available on V535's with the encoder option.

GET_POS	**Get command position**
Data/direction:	2/read
Encoding:	4Ah
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_POS returns the destination position that was set using the SET_POS command. It returns the double-buffered value (set by the host), that may or may not correspond to the active value, depending on whether a profile parameter update occurred. The returned position is a signed 32-bit number with units of steps.

GET_VEL	**Get command velocity**
Data/direction:	2/read
Encoding:	4Bh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_VEL returns the maximum velocity set using the SET_VEL command. It returns the double-buffered value (set by the host), that may or may not correspond to the active value, depending on whether a profile parameter update occurred. The returned velocity is an unsigned 32-bit number in $1/2^{16}$ format in units of steps/cycle.

GET_ACC	**Get command acceleration**
Data/direction:	2/read
Encoding:	4Ch
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_ACC returns the acceleration value set using the SET_ACC command. It returns the double-buffered value (set by the host), that may or may not correspond to the active value, depending on whether a profile update occurred. The returned value is either an unsigned 32-bit number in $1/2^{16}$ format with units of steps/cycle^2, or a signed 32-bit number in ½^{16} format with units if steps/cycle^2. The returned value is dependent on the type of profiling mode selected.

This command is used when the profiling mode is set to trapezoidal point to point or velocity contouring.

GET_MAX_ACC	**Get maximum acceleration**
Data/direction:	1/read
Encoding:	4Fh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_MAX_ACC returns the maximum acceleration value set using the SET_MAX_ACC command. It returns the double-buffered value (set by the host), that may or may not correspond to the active value, depending on whether a parameter update has occurred. The returned value is an unsigned number in $1/2^{16}$ format with units of steps/cycle^2.

This command is used only when the profiling mode is set to S-curve point to point.

GET_JERK	**Get command jerk**
Data/direction:	2/read
Encoding:	58h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_JERK returns the jerk value set using the SET_JERK command. It returns the double-buffered value (set by the host), that may or may not correspond to the active value, depending on whether a parameter update has occurred. The returned jerk value is an unsigned 32-bit number with $1/2^{32}$ scaling with units of steps/cycle^3.

GET_START_VEL	**Get starting velocity**
Data/direction:	2/read
Encoding:	6Bh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_START_VEL returns the starting velocity set using the SET_START_VEL command. The returned starting velocity is an unsigned 32-bit number using $1/2^{16}$ scaling with units of steps/cycle.

GET_TRGT_POS	**Return target position**
Data/direction:	2/read
Encoding:	1Dh

Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_TRGT_POS returns the current desired position being generated by the trajectory profile generator. This value represents the target position for the axis at the current cycle time. The current cycle time refers to the instant in time that the command was executed. This command is available in all trajectory profile modes. The returned value is a 32-bit signed number with units of steps. The range is from –1,073,741,824 to 1,073,741,823.

GET_TRGT_VEL	**Return target velocity**
Data/direction:	2/read
Encoding:	1Eh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_TRGT_VEL returns the current desired velocity value being generated by the trajectory profile generator. The returned value represents the target velocity for the axis at the current cycle time. The current cycle time refers to the instant in time that the command was executed. This command is available in all trajectory profile modes. The value returned for this command is a 32-bit number with units of steps/cycle represented in $1/2^{16}$ format. The range is from –1,073,741,824 to 1,073,741,823.

Parameter Update Commands

SET_TIME_BRK	**Set Break Point Mode to Time Based**
Data/direction:	none
Encoding:	17h
Axis acted on:	Current axis
Available on:	All axes
Double buffered:	no

SET_TIME_BRK sets the current breakpoint mode to be based on time. In this mode, the value loaded into the breakpoint register (SET_BRK_PNT command) will represent the number of cycles since the V535 was reset. After the SET_TIME_BRK command is executed, at each loop the break point value is compared to the current chipset time. If the values are equal all double buffered parameters are loaded into the active registers. See the GET_TIME command for additional information on chipset time. After this breakpoint condition has been satisfied, the breakpoint mode is reset and subsequent breakpoints on time must be reestablished using this command.

SET_POS_BRK Set Break Point Mode to Positive Target Position Based

Data/direction:	none
Encoding:	18h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_POS_BRK sets the current breakpoint mode to positive target position based. In this mode, the value loaded into the breakpoint register (SET_BRK_PNT command) will represent the axis position in steps. After the SET_POS_BRK command is executed, at each cycle the breakpoint value is compared against the current axis target position. If the target position is equal to or greater than the breakpoint register then all the double buffered parameters will be loaded into the active registers. After the breakpoint condition has been satisfied, the breakpoint mode is reset and subsequent breakpoints must be reestablished using the appropriate breakpoint mode.

SET_NEG_BRK Set Break Point Mode to Negative Target Position Based

Data/direction:	none
Encoding:	19h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_NEG_BRK sets the current breakpoint mode to positive target position based. In this mode, the value loaded into the breakpoint register (SET_BRK_PNT command) will represent the axis position in steps. After the SET_NEG_BRK command is executed, at each cycle the breakpoint value is compared against the current axis target position. If the target position is equal to or less than the breakpoint register then all the double buffered parameters will be loaded into the active registers. After the breakpoint condition has been satisfied, the breakpoint mode is reset and subsequent breakpoints must be reestablished using the appropriate breakpoint mode.

SET_ACTL_POS_BRK Set Break Point Mode to Actual Positive Target Position Based

Data/direction:	none
Encoding:	1Bh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_ACTL_POS_BRK sets the current breakpoint mode to positive actual position based. In this mode, the value loaded into the breakpoint register (SET_BRK_PNT command) will represent the axis position in steps. After the SET_ACTL_NEG_BRK command is executed, at each cycle the breakpoint value is compared against the current axis actual position. If the actual position is equal to or greater than the breakpoint register then all the double buffered parameters will be loaded into the active registers. After the breakpoint condition has been satisfied, the breakpoint mode is reset and subsequent breakpoints must be reestablished using the appropriate breakpoint mode.

This command is only available on V535s that include the encoder option.

SET_ACTL_NEG_BRK Set Break Point Mode to Actual Negative Target Position Based

Data/direction:	none
Encoding:	1Ch
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_ACTL_NEG_BRK sets the current breakpoint mode to positive actual position based. In this mode, the value loaded into the breakpoint register (SET_BRK_PNT command) will represent the axis position in steps. After the SET_ACTL_NEG_BRK command is executed, at each cycle the breakpoint value is compared against the current axis actual position. If the actual position is equal to or less than the breakpoint register then all the double buffered parameters will be loaded into the active registers. After the breakpoint condition has been satisfied, the breakpoint mode is reset and subsequent breakpoints must be reestablished using the appropriate breakpoint mode.

This command is only available on V535s that include the encoder option.

SET_MTN_CMPLT_BRK Set Break Point Mode to Motion Complete

Data/direction:	none
Encoding:	35h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_MTN_CMPLT_BRK sets the current breakpoint mode to motion complete. In this mode, the breakpoint condition is satisfied when the motion complete bit in the axis status word becomes true (axis motion is completed). This breakpoint is useful for initiating a new profile at the end of the current profile. Once the motion complete bit is true all double buffered parameters are loaded into the active registers. After the breakpoint condition has been satisfied, the breakpoint is reset and subsequent breakpoints must be reestablished using the appropriate breakpoint mode.

It should be noted that this command should not be executed while the Motion Complete bit is set. If so, an immediate breakpoint is issued.

SET_BRK_OFF	**Set Break Point Mode Off**
Data/direction:	none
Encoding:	6Dh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_BRK_OFF sets the breakpoint mode to ‘off’. Any breakpoint mode that had been previously set and is still active (the breakpoint has not occurred) is disabled via this command. After this command is executed, no additional breakpoints will occur until specifically enabled by another breakpoint set command.

SET_BRK_PNT	**Set Break Point Comparison Value**
Data/direction:	2/write
Encoding:	16h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_BRK_PNT sets the breakpoint comparison value used for other breakpoint commands. Its contents are interpreted based on the type of breakpoint mode set. For time based breakpoints (SET_TIME_BRK) the specified value is compared against the current chip set time at each cycle and the value specified is a 32-but number with units of cycles. For position based breakpoints (SET_POS_BRK, SET_NEG_BRK, SET_ACTL_POS_BRK, and SET_ACTL_NEG_BRK commands) the specified value is compared against the current axis target or actual position at each cycle and the value specified is a 32-bit number with units of steps.

UPDATE	**Immediately Update Parameters**
Data/direction:	none
Encoding:	1Ah
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

UPDATE immediately updates all double buffered parameters.

MULTI_UPDATE Immediately Update Parameters for Multiple Axis

Data/direction:	1/write
Encoding:	5Bh
Axis acted on:	Set by Data Word
Available on:	All axis
Double buffered:	no

MULTI_UPDATE immediately updates the double buffered parameter for 1 or more axis. For each updated axis, the axis behaves as if a separate UPDATE command had been executed for each axis. The data word associated with this command contains a mask bit for each axis. A "1" in the axis bit location indicates that the axis should be updated and bit locations set to "0" indicate no update. The following table shows the bit pattern for the data word.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	0	0	0	0	0	0	0	0	0	0	0	0	Axis 4	Axis 3	Axis 2	Axis 1

SET_AUTO_UPDATE_ON Set Automatic Profile Update On

Data/direction:	none
Encoding:	5Ch
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_AUTO_UPDATE_ON sets the automatic profile update mechanism on (enabled). After this command is executed, a breakpoint that is satisfied results in all of the double buffered parameters automatically being transferred to the active registers. Once set to this mode, the axis stays in this mode until explicitly commanded to disable this mode using the SET_AUTO_UPDATE_OFF command.

SET_AUTO_UPDATE_OFF Set Automatic Profile Update Off

Data/direction:	none
Encoding:	5Dh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_AUTO_UPDATE_OFF sets the automatic profile update mechanism off (disabled). After this command is executed, a satisfied breakpoint condition will not result in the double buffered parameters being updated. Once set to this mode, the axis stays in this mode until explicitly commanded to enable the update using the SET_AUTO_UPDATE_ON command. When in this mode (disabled), the only way to update profile parameters is through the UPDATE or MULTI_UPDATE commands.

GET_BRK_PNT Get Break Point Comparison Value

Data/direction:	2/read
Encoding:	57h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

GET_BRK_PNT returns the breakpoint comparison value set using the SET_BRK_PNT command. The returned 32-bit data value is in units of either cycles or steps, depending on the current breakpoint mode enabled.

Interrupt Processing Commands

SET_INTRPT_MASK Set Interrupt Mask

Data/direction:	1/write
Encoding:	2Fh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_INTRPT_MASK sets the interrupt mask so that interrupt sources can be individually masked on and off. When a non-masked interrupt (enabled) occurs in any axis, the Stepper Motor Interface Controller (SMIC) can generate an interrupt to VXI. This mask is used on conjunction with the Interrupt Control Register located in A16 address space. Once the SMIC generates an interrupt, no new interrupts will be generated until a RST_INTRPT command is executed. When the RST_INTRPT command is executed, the interrupt signal to the VXI interface is cleared and a new interrupt will be allowed through. The associated data word with this command is encoded as a field of bits, with each bit representing a possible interrupt source. A one (1) value in the mask bit location will cause the corresponding event to generate an interrupt. A zero (0) disables the event from generating an interrupt source. The following shows the bits defined for the data word associated with this command.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	0	0	0	0	0	0	0	0	Command Error	Neg Limit	Pos Limit	Motion Error	Pos Cap Rcvd	Updt Brk Rcvd	Pos Wrap	Motion Comp

Command Error **Bit 7**

Command Error is a write-only bit used to enable an interrupt source when the SMIC determines that a host communication sequence caused an error in the command. Setting this bit to a **"1"** enables the source and a **"0"** disables the source.

Negative Limit Switch **Bit 6**

Negative Limit Switch is a write-only bit used to enable an interrupt source when a negative (CCW) over-travel limit switch is activated. Setting this bit to a "1" enables the source and a "0" disables the source.

Positive Limit Switch **Bit 5**

Positive Limit Switch is a write-only bit used to enable an interrupt source when a positive (CW) over-travel limit switch is activated. Setting this bit to a "1" enables the source and a "0" disables the source.

Motion Error **Bit 4**

Motion Error is a write-only bit used to enable an interrupt source when the maximum position error set for a particular axis has been exceeded. Setting this bit to a "1" enables the source and a "0" disables the source.

Position Capture Received **Bit 3**

Position Capture Received is a write-only bit used to enable an interrupt source when the encoder index pulse has been captured. Setting this bit to a "1" enables the source and a "0" disables the source.

Update Breakpoint Reached **Bit 2**

Update Breakpoint Reached is a write-only bit used to enable an interrupt source when a breakpoint condition has been satisfied. Setting this bit to a "1" enables the source and a "0" disables the source.

Position Wrap **Bit 1**

Position Wrap is a write-only bit used to enable an interrupt source when the axis position wraps. Setting this bit to a "1" enables the source and a "0" disables the source.

Motion Complete **Bit 0**

Motion Complete is a write-only bit used to enable an interrupt source when an axis profile is complete. Setting this bit to a "1" enables the source and a "0" disables the source.

GET_INTRPT	**Return Status of Interrupting Axis**
Data/direction:	1/read
Encoding:	30h
Axis acted on:	Interrupting axis
Available on:	All axis
Double buffered:	-

GET_INTRPT returns the status of the axis that has generated the interrupt source from the Stepper Motor Interface Controller (SMIC). After this command is executed, the current axis number is not changed. Please refer to the GET_STATUS command description for details on the returned status word. If this command is executed and there is no interrupt source pending, the status for the current axis number is returned.

RST_INTRPT	**Reset Interrupting Condition Events**
Data/direction:	1/write

Encoding: 32h

Axis acted on: Interrupting axis

Available on: All axis

Double buffered: no

RST_INTRPT resets (clears) the interrupt source condition bits for the axis that has generated the interrupt. The data word associated with this command is encoded as a field of bits with each bit location referencing a possible interrupt source. For each bit location, a "1" written causes the status bit to remain unchanged and a "0" causes the corresponding event (bit) to be reset. The bit pattern for the data word is as follows:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	0	0	0	0	0	0	0	0	Command Error	Neg Limit	Pos Limit	Motion Error	Pos Cap Rcvd	Updt Brk Rcvd	Pos Wrap	Motion Comp

Please refer to the SET_INTRPT_MASK command description for additional information on the individual bits.

GET_INTRPT_MASK **Get Interrupt Mask**

Data/direction: 1/read

Encoding: 56h

Axis acted on: Current axis

Available on: All axis

Double buffered: no

GET_INTRPT_MASK returns the interrupt mask set by the SET_INTRPT_MASK command. This value is a bit encoded mask as described for the SET_INTRPT_MASK command,

Status/Mode Commands

CLR_STATUS **Clear all Event Bit Conditions**

Data/direction: none

Encoding: 33h

Axis acted on: Current axis

Available on: All axis

Double buffered: no

CLR_STATUS resets (clears) all of the event bit conditions for the axis (bits 7-0 of the status word). The Stepper Motor Interface Controller (SMIC) interrupt source line is not affected by this command. Since this command does not affect the interrupt source line, a RST_INTRPT command can be executed to clear the interrupt source. Please refer to the GET_STATUS command for a description of the status word bits.

RST_STATUS	**Reset Specific Event Bit Conditions**
Data/direction:	1/write
Encoding:	34h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

RST_STATUS reset (clears) the condition event bits for the current axis by using a data word mask. The data word is encoded as a field of bits with each bit location representing a possible event condition. For each bit location, a **"1"** written causes the status bit to remain unchanged and a **"0"** causes the status bit to be reset. The bit pattern for this command is as follows:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	0	0	0	0	0	0	0	0	Command Error	Neg Limit	Pos Limit	Motion Error	Pos Cap Rcvd	Updt Brk Rcvd	Pos Wrap	Motion Comp

Please refer to the SET_INTRPT_MASK command for the details on the individual bit locations.

GET_STATUS	**Get Axis Status Word**
Data/direction:	1/read
Encoding:	31h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_STATUS returns the status word for the current axis. Bits 7 through 0 of the status word are set by the Stepper Motor Interface Controller (SMIC) and must be reset by host command using either the CLR_STATUS, RST_STATUS, or RST_INTRPT commands. Bits 8,9,10,12 and 13 are continually maintained and cannot set or reset by the host. The bit pattern for the data word returned for this command is as follows:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	X	X	Current Axis Hi	Current Axis Lo	X	In Motion	Axis Active	Motor Active	Com-mand Error	Neg Limit	Pos Limit	Motion Error	Pos Cap Rcvd	Updt Brk Rcvd	Pos Wrap	Motion Comp

Current Axis Hi **Bit 13**

Current Axis Hi is used in conjunction with Current Axis Lo to determine the current axis being accessed. The binary combination of these bits reflects the actual axis number.

Current Axis Lo **Bit 12**

Current Axis Lo is used in conjunction with Current Axis Hi determines the current axis being accessed. The binary combination of these bits reflects the actual axis number.

In Motion **Bit 10**

This bit is returned as a "1" when the current axis is in motion. If the axis is at rest, this bit is returned as a "0".

Axis Active **Bit 9**

The Axis Active bit is returned as a "1" when the current axis is on and a "0" when the current axis is off.

Motor Active **Bit 8**

Motor Active is returned as a "1" when the motor is on and a "0" when the motor is off.

Command Error **Bit 7**

Command Error is returned as a "1" when a host communication sequence causes an error in the command. A "0" in this location indicates no command error.

Negative Limit Switch **Bit 6**

Negative Limit Switch is returned as a "1" a negative (CCW) over-travel limit switch is activated. This bit is returned as a "0" if no over-travel limit exists.

Positive Limit Switch **Bit 5**

Positive Limit Switch is returned as a "1" a positive (CW) over-travel limit switch is activated. This bit is returned as a "0" if no over-travel limit exists.

Motion Error **Bit 4**

Motion Error is returned as a "1" when the maximum position error set for a particular axis has been exceeded. A value of "0" is returned for this bit location is no error exists.

Position Capture Received **Bit 3**

Position Capture Received is returned as a "1" when the encoder index pulse has been captured. If no position capture has occurred, this bit is returned as a "0".

Update Breakpoint Reached **Bit 2**

Update Breakpoint Reached is returned as a "1" a breakpoint condition has been satisfied. A value if "0" if returned for this bit is the breakpoint has not yet been reached.

Position Wrap **Bit 1**

Position Wrap is returned as a "1" when the axis position wraps. A "0" value returned for this bit location indicates no wrap has occurred.

Motion Complete **Bit 0**

Motion Complete is returned as a "1" when an axis profile is complete. A value of "0" is returned if the axis is still in motion.

GET_MODE Get Axis Mode Word

Data/direction:	1/read
Encoding:	48h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_MODE returns the mode word for current axis. The bit pattern for the returned mode word is as follows:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Phase 2	Phase 1	Phase 0	Profile 1	Profile 0	Auto Update	Pulse Mode	RSVD	Error Stop	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

Phase 2-0 **Bits 15-13**

The Phase 2 through 0 bits reflects the current phase of an S-curve profile move.

Profile 1-0 **Bits 12-11**

The Profile1 through 0 bits reflect the current trajectory profile mode. The binary combination of these bits determine the mode as follows:

Profile 1	Profile 0	Trajectory Profile
0	0	Trapezoidal
0	1	Velocity Contouring
1	0	S-Curve

Table 5-1: V535 Trajectory Profile 1-0

Auto Update **Bit 10**

Auto Update is returned as a "1" when the automatic update feature is disabled. It is returned as a "0" when the automatic update feature is enabled.

Pulse Mode **Bit 9**

Pulse Mode is returned as a "1" when high-speed mode is selected and a "0" when low speed mode is selected.

Error Stop **Bit 7**

Error Stop is returned as a "1" when stop on motion error is enabled and a "0" when stop on motion error is disabled.

Pulse Generation Commands

SET_OUTPUT_STNDRD Set Pulse Generator Mode to Standard

Data/direction:	none
Encoding:	3Ch
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_OUTPUT_STNDRD sets the pulse generator output mode to standard. In this mode, the maximum pulse rate output by the Stepper Motor Interface Controller (SMIC) is 48.8 kilo-pulses/second. This command only affects the current axis.

SET_OUTPUT_HIGH Set Pulse Generator Mode to High Speed

Data/direction:	none
Encoding:	3Bh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_OUTPUT_HIGH sets the pulse generator output mode to high speed. In this mode, the maximum pulse rate output by the Stepper Motor Interface Controller (SMIC) is 1.55 mega-pulses/second. This command only affects the current axis.

MTR_ON Enable Pulse Generation Output

Data/direction:	none
Encoding:	43h

Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

MTR_ON enables pulse generation on the current axis. When pulse generation is enabled, pulse rate and direction information is generated by the trajectory generator and output onto the control lines. The control lines are either the Direction and Pulse pair or the Clockwise and Counter Clockwise pair. When this mode is disabled, no motor control signals are generated.

MTR_OFF	**Disable Pulse Generation Output**
Data/direction:	none
Encoding:	42h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

MTR_OFF disables pulse generation on the current axis. When pulse generation is disabled, pulse rate and direction information is immediately terminated. To re-enable the output signals, the MTR_ON command must be executed.

Encoder Commands

The following commands in this section are only applicable to V535s that incorporate the encoder option.

GET_ACTL_POS	**Return Actual Axis Position**
Data/direction:	2/read
Encoding:	37h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

GET_ACTL_POS returned the current encoder position of the current axis. The value returned for this command is up to date within one cycle time. The value is a 32 bit signed number with units of encoder counts.

SET_CAPT_INDEX	**Set Position Capture Trigger To the Index Signal**
Data/direction:	none
Encoding:	64h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_CAPT_INDEX sets the high-speed position register trigger source for the current axis to the index signal. When the index is used as the trigger source it is gated by the A and B quadrature signals.

GET_CAPT	**Return High Speed Capture Register**
Data/direction:	2/read
Encoding:	36h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_CAPT returns the current value of the high-speed position capture register as well as resets the capture hardware so that subsequent positions may be captured. The returned value is a 32 bit signed number with units of encoder counts.

SET_STEP_RATIO	**Set Number of Encoder Counts per Step**
Data/direction:	1/write
Encoding:	68h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_STEP_RATIO sets the ratio of encoder counts to output steps for the current axis that is used in conjunction with automatic stall detection. The specified ratio is a 16-bit unsigned number with a range of 0 to 32,767. The formula used to set this value is:

Ratio = (Ncounts/Nsteps)*256 (where Ncounts is the number of encoder counts per
Motor revolution and Nsteps is the number of output
Steps per motor revolution (12,800 for a 1.8 degree
Stepper and 3,200 for a 7.2 degree stepper)

Using this equation the resultant ratio must be an integer.

GET_STEP_RATIO	**Get Number of Encoder Counts per Step**
Data/direction:	1/read
Encoding:	6Fh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

GET_STEP_RATIO returns the ratio of encoder counts to output steps set using the SET_STEP_RATIO command. The returned value is a 16 bit unsigned number.

SET_AUTO_STOP_ON	**Enable Automatic Motor Shutdown**
Data/direction:	none
Encoding:	45h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_AUTO_STOP_ON enables the automatic profile generation shutdown when a motion error is detected. In this mode, the profile generation will be disabled (equivalent to MTR_OFF command) when a motion error occurs. Please refer to SET_POS_ERR for additional information on motion error. After an auto-stop has occurred, a MTR_ON command can be used to re-enable the motor.

SET_AUTO_STOP_OFF	**Disable Automatic Motor Shutdown**
Data/direction:	none
Encoding:	44h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_AUTO_STOP_OFF disables the automatic profile generator from shutdown due to a motion error. In this mode the profile generator is not halted when the error occurs.

SET_POS_ERR	**Set Position Error Limit**
Data/direction:	1/write
Encoding:	29h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_POS_ERR set the position error limit for the automatic stall detection feature. The error is specified as an unsigned 16-bit number with units of encoder counts. The range for this specification is from 0 to 32,767. At each chipset cycle the magnitude of the error calculated by the stall detector is compared with the specified position error set with this command. If the actual position exceeds the specified error limit, the motion error status bit is set. Also, if the axis has been set up to for automatic stop on motion error, the axis profile generator is disabled. Once this command is executed, the position limit checking occurs immediately. There is no requirement to execute an update command.

GET_POS_ERR	**Get Position Error Limit**
Data/direction:	1/read

Encoding:	55h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_POS_ERR returns the maximum position error that was set using the SET_POS_ERR command. The returned value is a 16 bit unsigned number with units of encoder counts.

GET_ACTL_POS_ERR **Return Current Position Error**

Data/direction:	1/read
Encoding:	60h
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	-

GET_ACTL_POS_ERR returns the current instantaneous position error of the current axis. The returned value represents the difference between the target position after a target motion, which has units of steps that has been converted into encoder counts using the step ratio parameters (set using the SET_RATIO command). The returned value is a signed 16-bit number with units of encoder count and ranges from –32,768 to 32,767.

Miscellaneous Commands

SET_ACTL_POS **Set Actual Axis Position**

Data/direction:	2/write
Encoding:	4dh
Axis acted on:	Current axis
Available on:	All axis
Double buffered:	no

SET_ACTL_POS sets the current actual position (in encoder counts) as well as the current target position (in steps) to the value specified with this command. The desired position is specified as a signed 32-bit number with an allowed range of –1,073,741,824 to 1,073,741,823. No update command is required for this command to take effect.

If this command is executed on a V535 with the encoder option included, the actual position error is set to zero.

SET_LMT_SENSE Set Limit Switch Bit Sense

Data/direction:	1/write
Encoding:	66h
Axis acted on:	Global (all axes)
Available on:	All axis
Double buffered:	-

SET_LMT_SENSE sets the interpretation of the limit switch input to the Stepper Motor Interface Controller (SMIC). The signal level interpretation of the positive (CW) and negative (CCW) limit switch inputs is programmable. A zero (0) in a bit location specifies that the input will be high true (active high). A one (1) in a bit location specified that the input will be low true (active low).

The Digital I/O Control Register located in A24 address space on the V535 can also be used to setup the logical sense of the limit switch inputs. The commands through the SMIC allow for this selection as well as the Digital I/O Control Register. The Digital I/O Control Register facilitates this setup by providing the additional Debouncer control bits in the same register.

The following shows the bit definitions for the SET_LMT_SENSE data word.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	0	0	0	0	0	0	0	0	Axis 4 Neg-Lim	Axis4 PosLim	Axis3 Neg-Lim	Axis3 PosLim	Axis2 Neg-Lim	Axis2 PosLim	Axis1 Neg-Lim	Axis1 PosLim

GET_LMT_SWTCH Get State of Over-Travel Limit Switches

Data/direction:	1/read
Encoding:	67h
Axis acted on:	Global (all axes)
Available on:	All axis
Double buffered:	-

GET_LMT_SWTCH returns the value of the limit switch input signals for all axes. The returned data word for this command has the following format:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	Axis 4 Neg-Lim	Axis4 PosLim	Axis3 Neg-Lim	Axis3 PosLim	Axis2 Neg-Lim	Axis2 PosLim	Axis1 Neg-Lim	Axis1 PosLim

Any bit location returned as a "1" indicates that the over-travel limit input is true (asserted). The values returned by this command are not affected by the SET_LMT_SENSE command or by the setup in the Digital I/O Control Register. These bits reflect the status of either true or false.

LMTS_ON	**Set Limit Switch Sensing On**
Data/direction:	none
Encoding:	70h
Axis acted on:	Global (all axes)
Available on:	All axis
Double buffered:	-

LMTS_ON turns on the limit switch sensing mechanism. LMTS_ON re-enables limit switch sensing whenever it has been disabled by using the LMTS_OFF command.

LMTS_OFF	**Set Limit Switch Sensing Off**
Data/direction:	none
Encoding:	71h
Axis acted on:	Global (all axes)
Available on:	All axis
Double buffered:	-

LMTS_OFF turns off the limit switch sensing mechanism. LMTS_OFF is used whenever it is necessary to disable limit sensing. This command only disables the automatic setting of the negative (CCW) and positive (CW) limit switch bits in the status word. It does not affect the status of these bits if they had been set prior to execution of this command, nor does it affect the GET_LMT_SWTCH command.

RESET	**Reset the Stepper Motor Interface Controller**
Data/direction:	none
Encoding:	39h
Axis acted on:	Global (all axes)
Available on:	All axis
Double buffered:	no

RESET is used to reset the entire chipset that makes up the Stepper Motor Interface Controller (SMIC). This command performs the same function as if a hardware reset was executed. At the end of a RESET operation, the SMIC is in a default state with the following status:

Condition	Initial Value
All Actual Axis Positions	0
All Capture Registers	0

Condition	Initial Value
All Event Conditions	Cleared
SMIC Interrupt Source	Cleared
All Interrupt Masks	Disabled
All Profile Modes	Trapezoidal
All Profile Parameter Values	0
All Breakpoint Comparison Values	0
Automatic Update	Enabled (On)
Capture Input Mode	Index
Limit Switch Sensing	Enabled (On)
Limit Switch Sense	0 (Active High)
All Pulse Generator Modes	Standard
All Pulse Rates	0
Current Axis Number	1
All Current Actual Positions	0
All Step ratios	0
All Actual Position Errors	0
All Motor Status	On

Table 5-2: V535 Reset Status

GET_VRSN **Return SMIC (chipset) Software Information**

Data/direction: 1/read

Encoding: 6Ch

Axis acted on: Global (all axes)

Available on: All axis

Double buffered: -

GET_VRSN returns various information regarding the chipset part number and software revision. The following shows the bit definitions for the data returned from this command.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Gen 1	Gen 0	Ax 2	Ax 1	Ax 0	Pn 2	Pn 1	Pn 0	Dv 2	Dv 1	Dv 0	Major 1	Major 0	Minor 2	Minor 1	Minor 0

Generation 1 and 0 **Bits 15-14**

These two bits reflect the generation of the chipset.

Axes Supported 2 through 0 **Bits 13-11**

These 3 bits return the number of axes supported by the chipset.

Part Number 2 through 0 **Bits 10-8**

These 3 bits encode the part number for the chipset. The following table shows the data returned for these bit locations and the chipset part numbers that they describe.

Part Number Bits 2 through 0	Chipset Part Number
0	MC1400 Series
1	MC1401 Series
2	MC1231 Series
3	MC1451 Series
4	MC1451 Series

Table 5-3: V535 Bit Locations and Chipset Part Numbers

GET_TIME	**Return SMIC (chipset) Time**
Data/direction:	2/read
Encoding:	3Eh
Axis acted on:	Global (all axes)
Available on:	All axis
Double buffered:	-

GET_TIME returns the current system time, expressed as the number of cycles since chipset power was applied. The chipset clock starts at 0 after a power on cycle or reset and will count continually, wrapping from a value of 4,294,967,296 to 0. The returned value is a 32-bit number with units of cycles.

Stepper Motor Indexer Command Summary

Command Mnemonic	Hex Code	Axes acted on	# data words / direction	Description
Axis Control Commands				
SET_1	01	Set by command	1/read	Set current axis number to 1
SET_2	02	Set by command	1/read	Set current axis number to 2
SET_3	03	Set by command	1/read	Set current axis number to 3
SET_4	04	Set by command	1/read	Set current axis number to 4
SET_I	08	Interrupting Axis	1/read	Set current axis number to interrupting axis
Profile Generation Commands				
SET_PRFL_S_CRV	0B	Current Axis	0	Set profile mode to S-curve
SET_PRFL_TRAP	09	Current Axis	0	Set profile mode to trapezoidal point to point
SET_PRFL_VEL	0A	Current Axis	0	Set profile mode to velocity-contouring
SET_POS	10	Current Axis	2/write	Set command position
SET_VEL	11	Current Axis	2/write	Set command velocity
SET_ACC	12	Current Axis	2/write	Set command acceleration
SET_MAX_ACC	15	Current Axis	1/write	Set maximum acceleration (S-curve profile only)
SET_JERK	13	Current Axis	2/write	Set command jerk
SET_START_VEL	6A	Current Axis	2/write	Set starting velocity
STOP/CLR_PRFL	46	Current Axis	0	Abruptly stop current axis motion
SMOOTH_STOP	4E	Current Axis	0	Smoothly stop current axis motion
SYNH_PRFL	47	Current Axis	0	Set target position equal to actual position

Command Mnemonic	Hex Code	Axes acted on	# data words / direction	Description
GET_POS	4A	Current Axis	2/read	Get command position
GET_VEL	4B	Current Axis	2/read	Get command velocity
GET_ACC	4C	Current Axis	2/read	Get command acceleration
GET_MAX_ACC	4F	Current Axis	1/read	Get maximum acceleration (S-curve only)
GET_JERK	58	Current Axis	2/read	Get command jerk
GET_START_VEL	6B	Current Axis	2/read	Get starting velocity
GET_TRGT_POS	1D	Current Axis	2/read	Get current target position
GET_TRGT_VEL	1E	Current Axis	2/read	Get current target velocity
Parameter Update				
SET_TIME_BRK	17	Current Axis	0	Set breakpoint mode to time
SET_POS_BRK	18	Current Axis	0	Set breakpoint mode to positive target position
SET_NEG_BRK	19	Current Axis	0	Set breakpoint mode to negative target position
SET_ACTL_POS_BRK	1B	Current Axis	0	Set breakpoint mode to positive actual position
SET_ACTL_NEG_BRK	1C	Current Axis	0	Set breakpoint mode to negative actual position
SET_MTN_CMPLT_BRK	35	Current Axis	0	Set breakpoint mode to motion complete
SET_EXT_BRK	5E	Current Axis	0	Set breakpoint mode to external (Not supported)
SET_BRK_OFF	6D	Current Axis	0	Set breakpoint mode off
SET_BRK_PNT	16	Current Axis	2/write	Set breakpoint comparison mode
UPDATE	1A	Current Axis	0	Immediate parameter update
MULTI_UPDATE	5B	Set by Mask	1/write	Multiple axis immediate parameter update
SET_AUTO_UPDATE_ON	5C	Current Axis	0	Set automatic profile update on
SET_AUTO_UPDATE_OFF	5D	Current Axis	0	Set automatic profile update off
GET_BRK_PNT	57	Current Axis	2/read	Get breakpoint comparison value

Command Mnemonic	Hex Code	Axes acted on	# data words / direction	Description
SMIC Interrupt Processing				
SET_INTRPT_MASK	2F	Current Axis	1/write	Set interrupt mask
GET_INTRPT	30	Interrupting Axis	1/read	Get status of interrupting axis
RST_INTRPT	32	Interrupting Axis	1/write	Reset interrupting events
GET_INTRPT_MASK	56	Current Axis	1/read	Get interrupt mask
Status/Mode				
CLR_STATUS	33	Current Axis	0	Reset status of current axis
RST_STATUS	34	Current Axis	1/write	Reset events for current axis
GET_STATUS	31	Current Axis	1/read	Get axis status word
GET_MODE	48	Current Axis	1/read	Get axis mode word
Pulse Generation				
SET_OUTPUT_STNDRD	3C	Current Axis	0	Set pulse generation mode to standard
SET_OUTPUT_HIGH	3B	Current Axis	0	Set pulse generation mode to high speed
MTR_ON	43	Current Axis	0	Enable pulse output
MTR_OFF	42	Current Axis	0	Disable pulse output
Miscellaneous				
SET_ACTL_POS	4D	Current Axis	2/write	Set axis position
SET_LMT_SENSE	66	Global	1/write	Set limit switch bit sense
GET_LMT_SWTCH	67	Global	1/read	Get state of limit switches
LMTS_ON	70	Global	0	Set limit switch sensing on
LMTS_OFF	71	Global	0	Set limit switch sensing off
GET_HOME	05	Global	1/read	Get state of home switches (not supported)
RESET	39	Global	0	Reset chipset
GET_VRSN	6C	Global	1/read	Get chipset software version information
GET_TIME	3E	Global	2/read	Get current chip set time (# cycles)

Command Mnemonic	Hex Code	Axes acted on	# data words / direction	Description
Encoder Option Commands				
GET_ACTL_POS	37	Current Axis	2/read	Get current actual axis location
SET_STEP_RATIO	68	Current Axis	1/write	Set number of encoder counts per step
GET_STEP_RATIO	6F	Current Axis	1/read	Get number of encoder counts per step
SET_AUTO_STOP_ON	45	Current Axis	0	Set auto stop on motion error on
SET_AUTO_STOP_OFF	44	Current Axis	0	Set auto stop on motion error off
SET_POS_ERR	29	Current Axis	1/write	Set maximum position error limit
GET_POS_ERR	55	Current Axis	1/read	Get maximum position error limit
GET_ACTL_POS_ERR	60	Current Axis	1/read	Get actual position error
SET_CAPT_INDEX	64	Current Axis	0	Set position capture trigger source to the index signal
GET_CAPT	36	Current Axis	2/read	Return high speed capture register

Executing Stepper Motor Indexer Commands

As an example, assume it is desired to execute a Trapezoidal Profile move with the V535. For this example, the axis to be manipulated is axis 1. Axis 1 is to be moved in a clockwise direction for one revolution. After the move is complete, an interrupt is to be generated informing the host that the requested operation is complete. The requirements of the move are as follows:

Stepper Motor number of steps/revolution	400
Starting Velocity for the motor	0
Maximum Velocity for the motor	10,000
Acceleration for the motor	1,000
Position specification for the move	400
Interrupt Request Level	7

Since the motor has 400 steps per revolution, the V535 needs to provide a pulse train of 400 steps.

First, to implement a trapezoidal profile, we need to know:

1. The V535 axis we wish to use.
2. The final position of the motor.
3. The desired maximum velocity of the motor.
4. The desired acceleration of the motor.
5. The starting velocity of the motor.

Initially, to setup the trapezoidal move, the aforementioned parameters must be loaded into the V535. This can be accomplished through use of the *ksv535_TrapProfileSet* command. The values passed to this command include the axis number, the final motor position, the maximum desired velocity, the acceleration, and the starting velocity. Once this command is performed, the loaded values are not utilized until a parameter update occurs. The update command is performed through the ks*v535_SmicExecute* command with the command field set to *UPDATE.*

Before the move operation is executed, some setup must occur to enable the V535 to generate the interrupt once the move is complete. There are several setup requirements as follows:

VXI Interrupt Request Level

VXI Interrupt Mask

Interrupt Mask for the Stepper Motor Interface Controller

The V535 axis we wish to use

The VXI Interrupt Request Level and the VXI Interrupt Mask are configured by using the Interrupt Control Register located in A16 address space on the V535. The Interrupt Mask for the Stepper Motor Interface Controller and the desired axis are configured through various Stepper Motor Indexer commands. However, this setup can be achieved through the *ksv535_InterruptEnable* command. The values passed to this command are the VXI Interrupt Mask, the VXI Interrupt Level, the axis on which to enable the demands, and the Stepper Motor Interface Controller Interrupt Mask.

The following discussion refers to the 'C' source code listing following this description.

Once these values are determined we need to know what devices you have available. We do this with the *ksv535_getDeviceList* command. This command returns the number of devices found along with their slot numbers and logical addresses. Next, using the *ksv535_init* command, we initialize the first V535 found. This command also returns the unique session number, hereafter referred to as *vi,* that will be used in all of the visa and plug and play calls. Now we need to enable the V535 to interrupt when the motion is complete and install a handler to service these interrupts. This is done with three functions. First we call *ksv535_InterruptEnable*. In this call we specify that we are enabling the V535 to interrupt on IRQ7 when motion is complete.

(Note that the stepper motor indexer is issuing the interrupt. We need to specify that it interrupts on motion complete. See the Stepper Motor Indexer Command (SMIC) SET_INTRPT_MASK in chapter 5 of the V535 manual for more interrupt sources.)

Next we specify the handler function through *viInstallHandler*. For our purposes this function is *handlerFunction*. Now we enable events to occur with the *viEnableEvent* command.

Now we are ready to load the desired motor parameters. This is done through the *ksv535_TrapProfileSet* command. This command will load the parameters in the first buffer.

This does not initiate motion. To initiate motor motion we need to issue the *UPDATE* command. This is done through the *ksv535_SmicExecute* command. Now we wait.

Once motion is complete the V535 will issue an interrupt and *handlerFunction* will be called.

Once inside *handlerFunction* we make the interrupting axis the active axis by issuing the *SET_I* command through the *ksv535_SmicExecute* command. We then clear the motion complete flag by issuing a *RST_STATUS* command through *ksv535_SmicExecute.* Next we clear the interrupt source condition for this axis with the *RST_INTRPT* SMIC. This allows this condition to occur again. We then let the waiting code know that the motion has completed.

When we receive notification of motion complete we are done. We are free to disable interrupts and discard any pending events. This is done with the *ksv535_InterruptEnable, viDisableEvent, viDiscardEvents,* and *viUninstallHandler* functions. We then close the session to the V535 with the *viClose* command and we a done.

A copy of the following code has been included in the *examples* directory of the V535 plug and play driver.

```
/******************************************************************************
This program will set up the V535 and execute a Trapezoidal profile.
The V535 will signal, through an interrupt, that motion is complete.
******************************************************************************/
#include <visa.h>
#include "ksv535.h"

/***************************************************************************
Defines
***************************************************************************/

#define MAX_V535S              12     /*Max number of V535's in a crate*/
#define MESG_LEN               256    /*For status and error messages*/

/***************************************************************************
Global Variables
***************************************************************************/

ViSession   vi;            /* unique session id */
ViChar      message[MESG_LEN];
struct      ksv535_data  *deviceData;

ViBoolean blnDone;  /* Flag used to signal motion complete */

/***************************************************************************
Function Prototypes
***************************************************************************/
ViStatus _VI_FUNCH handlerFunction(  ViSession vi,
                                     ViEventType etype,
                                     ViEvent context,
                                     ViAddr unserHandle);

ViInt32 ksv535_interrupt_example (ViUInt16 usAxis, ViInt32 slPosition,
                                  ViUInt32 ulVelocity, ViUInt32 ulAccel,
                                  ViUInt32 ulStartVel_Jerk);

void main()
{
        ViStatus status;
        ViInt32  Position;
        ViUInt32 Velocity,
                Acceleration,
                StartVel_Jerk;
        ViUInt16 Axis;

        Axis = 1;           //1 - 8
        Position = 400;     //Destination position
        Velocity = 10000;   //Max Velocity
        Acceleration = 1000; //Acceleration
        StartVel_Jerk = 0;  //Starting velocity for trapezoidal profile
                            //Jerk for S-Curve profile.
        status = ksv535_interrupt_example(Axis,
                                         Position,
                                         Velocity,
                                         Acceleration,
                                         StartVel_Jerk);
        if(status != 0)
        {
                printf("error in example code\n");
        }

}
```

```
ViInt32 ksv535_interrupt_example (ViUInt16 usAxis, ViInt32 slPosition,
                                  ViUInt32 ulVelocity, ViUInt32 ulAccel,
                                  ViUInt32 ulStartVel_Jerk)
{
        ViBoolean       blnIdQuery,                     /* Enable ID query?  */
                        blnReset,                       /* Enable and reset? */
                        blnEnable = VI_TRUE,
                        blnSumCheck = VI_FALSE,
                        blnGroup = GROUP_A;

        ViInt16         listLength,             /* Max number of V200's in a crate */
                        numFound,               /* Number of V200s actually found  */
                        laList[256],            /* Logical Address list array      */
                        slotList[256];          /* Physical location list array    */

        ViUInt32     ulWriteData = 0,
                     ulReadData;

        ViStatus        status;                 /*Error messages*/

        ViChar          rsrcName[MESG_LEN];             /*Unique resource name*/

        /*************************************************************************
        Locate the first V535
        *************************************************************************/

        listLength  = MAX_V535S;

        status = ksv535_getDeviceList(listLength, /*Max number of V200's in Crate*/
                                      &numFound,  /*Number of V200s actually found*/
                                      slotList,   /*Physical location list array*/
                                      laList);    /*Logical Address list array*/
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        /*************************************************************************
        Initialize first V535 found
        *************************************************************************/

        sprintf(rsrcName, "VXI0::%d::INSTR", laList[0]);/*Generate resource name*/
        blnIdQuery = VI_TRUE;                            /*Enable ID query?*/
        blnReset = VI_TRUE;                              /*Enable reset?*/

        status = ksv535_init( rsrcName,    /*Resource name*/
                              blnIdQuery,  /*Enable query id? */
                              blnReset,    /*Enable reset? */
                              &vi);        /*Unique session number*/
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        /*************************************************************************
        Enable Interrupts from SMIC indicating motion complete
        *************************************************************************/
        blnEnable = VI_TRUE;
        status = ksv535_InterruptEnable(vi,

                                        blnEnable, //Enable Interrupts
                                        KSV535_INT_ENABLE | KSV535_INT_SMIC_A ,//Interrupts from SMIC A
```

```
                                   KSV535_INT_IRQ7, //Interrupt on IRQ 7
                                   1,     //enable axis 1 to cause the interrupt
                                   KSV535_SMIC_MOTION_COMP); //interrupt when motion complete
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        status = viInstallHandler(vi,                                  /*Unique session number*/
                                  VI_EVENT_VXI_SIGP,                   /*VXI Event: bus signal*/
                                  handlerFunction,   /*Prototype name of callback func*/
                                  deviceData);       /*Refernce the handler*/

        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        status = viEnableEvent(      vi,                    /*Unique session number*/
                                     VI_EVENT_VXI_SIGP,     /*VXI Event: bus signal*/
                                     VI_HNDLR,              /*VISA Handler, or Queue (HANDLER)*/
                                     VI_NULL);              /*Should be set to VI_NULL*/

        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        if(usAxis < AXIS_5)
        {
                blnGroup = GROUP_A;
        }
        else
        {
        blnGroup = GROUP_B;
}


        status = ksv535_TrapProfileSet(vi,
                                       usAxis,
                                       slPosition,
                                       ulVelocity,
                                       ulAccel,
                                       ulStartVel_Jerk);
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }
        blnDone = VI_FALSE;
        status =  ksv535_SmicExecute(vi, UPDATE,&ulReadData,0, blnGroup,0);
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        while (blnDone == VI_FALSE)
        {
                printf("Motor in motion\r");
        }
        printf("Motion complete\n");

        /************************************************************************
        Disable interrupt and discard pending events
        ************************************************************************/

        blnEnable = VI_FALSE;  /*Disable interrupt*/

        status =  ksv535_InterruptEnable(vi,
                          blnEnable,
                          KSV535_INT_ENABLE | KSV535_INT_SMIC_A,
                          KSV535_INT_IRQ7,
                          1,
                          KSV535_SMIC_MOTION_COMP);
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        status = viDisableEvent(vi,                           /*Unique session number*/
                        VI_ALL_ENABLED_EVENTS,       /*Disable all enabled events*/
                        VI_ALL_MECH);                /*On all mechanisms*/
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        status = viDiscardEvents(vi,                          /*Unique Session Number*/
                          VI_ALL_ENABLED_EVENTS,     /*Discard all enabled events*/
                          VI_ALL_MECH);              /*On all mechanisms*/
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        status = viUninstallHandler(vi,                               /*Unique Session Number*/
                             VI_EVENT_VXI_SIGP,                /*Event on VXIbus signal*/
                             (ViAddr)handlerFunction,          /*Func called by event*/
                             deviceData);                      /*User handle*/
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        /************************************************************************
        Close vi session
        ************************************************************************/

        status = ksv535_close(vi);    /*Unique session number*/
        if(status < VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }
        return 0;

}//end main

/*****************************************************************************
Handler function called when an interrupt occurs
```

```
**********************************************************************************/

ViStatus _VI_FUNCH handlerFunction(   ViSession vi,
                                      ViEventType etype,
                                      ViEvent context,
                                      ViAddr unserHandle)
{
        ViUInt32  ulWriteData,
                  ulReadData;
        ViBoolean blnGroup = GROUP_A,
                  blnSumCheck = VI_FALSE,
                  blnEnable;
        ViStatus  status;

        status = ksv535_SmicExecute(vi,SET_I, &ulReadData, ulWriteData,
                                    blnGroup,blnSumCheck);
        if(status != VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        ulWriteData = 0xFE; //clear the motion complete bit
        status = ksv535_SmicExecute(vi,RST_STATUS, &ulReadData, ulWriteData,
                                     blnGroup,blnSumCheck);
        if(status != VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        status = ksv535_SmicExecute(vi,RST_INTRPT, &ulReadData, ulWriteData,
                                    blnGroup,blnSumCheck);
        if(status != VI_SUCCESS)
        {
                ksv535_error_message(vi, status, message);
                printf("%s\n",message);
                return(1);
        }

        //Let the user know that motion is complete
        blnDone = VI_TRUE;
        return VI_SUCCESS;
}/*END handlerFunction*/
```

Chapter 6: Appendix:About KineticSystems

KineticSystems Corporation designs, produces and markets high-performance data acquisition and control systems to a broad range of customers in aerospace, defense, automotive, scientific and other industrial markets.

The Leader in the Delivery of High-performance CAMAC-based Products

The company was founded in 1970 to develop and manufacture interface modules and associated products based on the international CAMAC standard. CAMAC, an acronym for Computer Automated Measurement and Control, is a set of specifications developed by a committee of the government-sponsored research laboratories (the NIM Committee). The committee's goal was to provide standardized modular building blocks for configuring a wide range of data acquisition and control systems. CAMAC, the first open-system real-time input/output (I/O) specification, later was accepted as a standard by the Institute of Electrical and Electronic Engineers (IEEE STD 583). We soon became the recognized leader in delivering high-performance CAMAC-based products. We have retained that leadership position.

Innovation with the CAMAC Serial Highway

As the need for large distributed data acquisition and control systems grew, the NIM Committee produced specifications for the CAMAC Serial Highway to allow communication between a host computer and CAMAC I/O chassis over some distance. In 1975, we delivered the first CAMAC Serial Highway computer interface. We soon were recognized as the leader in the delivery of Serial Highway system components. Serial Highway innovations included a block-mode protocol that increased data throughput by a factor of 10 and fiber-optic highway interfaces that allow the CAMAC chassis to be separated by up to 2 kilometers at full data rate. We continue to be an innovator in the field of high-performance data acquisition.

H•TMS, a Turn-key Testing Solution, Added to Our Product Range

In 1991, we purchased the H•TMS (High-performance Test Management System) product line. H•TMS is a modular set of microprocessor-based hardware and software components that delivers functions usually found in several instruments and a computer. H•TMS increases testing productivity and provides solutions to key problems encountered by test engineers and managers. The addition of this product line provides the answer for customers who need a turnkey testing solution.

A Major Player in VXIbus, a Rapidly Growing Interface Standard

VXIbus is a standard (now IEEE STD 1155) developed by the major instrumentation manufacturers to allow customers to move from chassis-type instruments to computer-interfaced modular building blocks. In 1992, we embraced the relatively new VXIbus standard. Using our extensive CAMAC experience, we soon produced 35 VXI modules for data acquisition and control. As the VXI market has grown to several hundred million dollars, we continue our innovations with additional high-performance products. This includes the development of a set of products called the Grand Interconnect™. These products allow VXI chassis to be distributed on a fiber-optic highway. CAMAC and mixed VXI/CAMAC chassis are also supported. We are an active member of the VXI *plug&play* Systems Alliance, an organization that promotes standards that make VXI easier to configure and to use.

KineticSystems Today

Our headquarters and factory facilities in Lockport, Illinois, have grown to 70,000 square feet. We also have an operation in Englewood, Colorado, and six domestic sales offices, including the Lockport facility. We have distributors in 17 other countries. VXI, CAMAC and H•TMS continue to be our major product lines.

Today, an increasing number of customers are demanding the delivery of data acquisition and control solutions, not just products. For many years we have provided software drivers for our products to make them easier to use. We have an active program to develop VXI *plug&play* instrument drivers for our range of VXI modules. We have developed a high-performance software application program called Reality®. Distributed VXI and/or CAMAC systems can be configured using this software. By using UNIX workstations and powerful I/O control computers, a high I/O performance can be achieved that is not available with any other general-purpose software package. Additionally, to achieve more complete solutions, we provide integration services.

KineticSystems is ideally suited to meet customers' needs as we approach the twenty-first century. We have extensive experience in the design, manufacture and delivery of high-performance data acquisition products and systems. Although the demand for standards-based products has only recently become a high priority, we have the experience with such products since 1970.

Ways to contact us:

✉

KineticSystems Company, LLC
900 N. State Street
Lockport, IL 60441-2200

☎

Phone: 1-800-DATA NOW (1-800-328-2669)
(815) 838-0005
Fax: (815) 838-4424

💻

E-mail: mkt-info@kscorp.com
tech-serv@kscorp.com
sales@kscorp.com
Web: http://www.kscorp.com

Chapter 7: Warranty

KineticSystems warrants its standard hardware products to be free of defects in workmanship and materials for a period of one year from the date of shipment to the original end user. KineticSystems warrants its software products to conform to the software description applicable at the time of purchase for a period of ninety days from the date of shipment. Products purchased for resale by KineticSystems carry the original equipment manufacturer's warranty.

KineticSystems will, at its option, either repair or replace products that prove to be defective in materials or workmanship during the warranty period.

Transportation charges for shipping products to KineticSystems are prepaid by the purchaser, while charges for returning the repaired product to the purchaser, if located in the United States, are paid by KineticSystems. Return shipments are made by UPS, where available, unless the purchaser requests a premium method of shipment at his expense. The selected carrier is not the agent of KineticSystems, and KineticSystems assumes no liability relating to the services provided by the carrier.

The product warranty may vary outside the United States or Switzerland and does not include shipping, customs clearance or any other charges. Consult your local authorized representative for more information regarding specific warranty coverage and shipping details.

Product specifications and descriptions in this document subject to change without notice.

KineticSystems specifically makes no warranty of fitness for a particular purpose or any other warranty either expressed or implied, except as is expressly set forth herein. This warranty does not cover product failures created by unauthorized modifications, product misuse or improper installation.

Products are not accepted for credit or exchange without prior written approval. If it is necessary to return a product for repair replacement or exchange, a Return Authorization (RA) Number must first be obtained from the Repair Service Center before shipping the product to KineticSystems.

Please take the following steps if you are having a problem and feel you may need to return a product for service:

1. Contact KineticSystems and discuss the problem with a Technical Service Engineer.
2. Obtain a Return Authorization (RA) Number.
3. Initiate a purchase order for the estimated repair charge if the product is out of warranty.
4. Include with the product a description of the problem and the name of the technical contact person at your facility.
5. Ship the product prepaid with the RA Number marked on the outside of the package to:

KineticSystems Company, LLC
Repair Service Center
900 North State Street
Lockport, IL 60441
Telephone: (815) 838-0005
Fax: (815) 838-4424

Chapter 8: Feedback

The purpose of this manual is to provide you with the information you need to make the V535 as easy as possible to understand and use. It is very important that the information is accurate, understandable and accessible. To help us continue to make this manual as "user friendly" as possible, we hope you will fill out this form and Fax it back to us at (815) 838 4424. Or mail a copy to KineticSystems Company, LLC, 900 N. State, Lockport, IL 60441. Your input is very valuable.

Please rate each of the following.

The information in this manual is:

	Yes									No
Accurate	10	9	8	7	6	5	4	3	2	1
Readable	10	9	8	7	6	5	4	3	2	1
Easy to find	10	9	8	7	6	5	4	3	2	1
Well organized	10	9	8	7	6	5	4	3	2	1
Sufficient	10	9	8	7	6	5	4	3	2	1

We would appreciate receiving any thoughts you have about how we can improve this user's manual:

(Include additional sheets if needed)

Name **Phone**

Company