KOLLMORGEN

Motion Technologies Group

VECTOR**STAR** (VSA/VSP/VSL) Installation and User's Manual

Manual MVS000H

February 1999

VECTOR**STAR** (VSA/VSP/VSL)
Installation and User's Manual

MVS000H

Printed in the United States of America.

NOTICE:

Dangerous voltages, currents, temperatures, and energy levels exist in this product and in the associated servo motor(s). Extreme caution should be exercised in the application of this equipment. Only qualified individuals should attempt to install, set-up, and operate this equipment. Ensure that the motor, drive, and the end-user assembly are all properly grounded per NEC requirements.

KOLLMORGEN

Motion Technologies Group

INDUSTRIAL DRIVES
201 Rock Road
Radford, VA 24141
Phone: 540/639/2495
Fax: 540/731/0847

TECHNICAL MANUAL CONFIGURATION

(INSTALLATION AND USER'S MANUAL MVS000H)

PAGE NO.	DESCRIPTION	ISSUE NO.*
—	Title Page	0
—	Technical Manual Configuration	0
—	Configuration Table	0
—	Customer Response	0
—	Copyright Page	0
—	Foreword	0
—	How to Use This Manual	0
i-ii	Table of Contents	0
iii / (iv blank)	List of Figures	0
v / (vi blank)	List of Tables	0
vii / (viii blank)	List of Drawings	0
1-1 — 1-16	Text Pages	0
2-1 — 2-26	Text Pages	0
3-1 — 3-10	Text Pages	0
4-1 — 4-4	Text Pages	0
5-1 — 5-6	Text Pages	0
A-1 / (A-2 blank)	Appendix A Warranty Information	0
B-1 — B-22	Appendix B Drawings	0
C-1 — C-12	Appendix C Error Codes	0
D-1 — D-2	Appendix D Regional Sales Offices	0
E-1 / (E-2 blank)	Appendix E On-line Reference Guide	0
Glossary-i — xiv	Glossary	0
Index-i — iv	Index	0

* Zero in this column indicates an original page

CONFIGURATION TABLE

(INSTALLATION AND USER'S MANUAL MVS000H)

RECORD OF REVISIONS

ISSUE NO. (Revised)	DATE	CHANGED PAGES/BRIEF DESCRIPTION OF CHANGE	CHANGE NO.
		December 19, 1996	Original Release
1	13 June 97	Replaces issue dated 19 December 96	--
2	14 Aug. 97	Replaces issue dated 13 June 97	--
3	5 Feb. 99	Replaces issue dated 14 August 97	--

FOREWORD

Commitment to quality at Kollmorgen is our first priority. In all aspects of our business: research, development, product design and customer service, we strive to guarantee total quality. This pledge is founded on a solid history of innovative technological achievements dating back to 1948. One of the finest tributes to that achievement is now on display at the Smithsonian, the first stellar inertial navigation system developed by Dr. Charles Stark Draper. This system contains the first models of torque motors built by the founding organization of Kollmorgen. During the period of 1948 to 1960, our "firsts" in the industry numbered more than a dozen; they ranged from the simple but invaluable (such as the direct-drive DC torque motor and movie theater projection motors) to the exotic: submarine periscope drive motors for the U.S. Navy, electric drives, Curtis Wright electric brake coils, and numerous other innovations.

For more than a decade, Kollmorgen has continued to enhance its sophisticated engineering solutions to pioneer new product development.

The results of these and other efforts have encouraged some of the most significant innovations in the servo industry. We developed the application of servo motors and drives in the Machine Tool market. We were the first with water-cooled servos, the integral brake, the flux forcing concept, and the brushless motor. We developed the electronically commutated electric car motor. Kollmorgen pioneered rare earth magnet development for the servo motor industry.

Between 1974 and 1980, Kollmorgen Drives continued to lead the industry in servo application innovations. Our commitment to engineering excellence never waivered. In fact, that commitment grew stronger with the development of brushless submarine and submersible motors (visiting the Titanic graveyard), multi-axis electronic drives and antenna pedestal drives (delivering unprecedented accuracy and revolutionizing the entire industrial automation process).

The 1980's brought continued advancements in technology and penetration of new markets requiring precise motion control. Already in the fifth generation of brushless products, Kollmorgen continues to lead the way with digital servo positioning capability and our newest offering, the VECTOR**STAR** Series. Once again, we are setting the standards that others only hope to duplicate. Recently acknowledged by the Frost and Sullivan Foundation, a leading market specialist in the motion control industry, Kollmorgen Corporation continues to rank first in servo technology.

Other achievements? Yes, too many in fact to mention. Each achievement stands as a testimony to the commitment to quality and excellence in design technology. This constancy of purpose is unyielding in an era of rapidly changing technology.

HOW TO USE THIS MANUAL

INTRODUCTION

This manual is designed to help you properly install and use a VECTOR**STAR** Spindle Drive or Servo System. You do not have to be an expert in motion control to operate the system; however, this manual does assume you have the fundamental understanding of basic electronics and motion control concepts. Many of these are explained in the glossary of this manual.

VECTOR**STAR** is a programmable motion control device. An understanding of computer programming techniques will be beneficial to all users. For applications that require complex programs, a professional programmer should be consulted.

RECOMMENDATIONS

It is recommended that you read this entire manual before you attempt to install your VECTOR**STAR** system so you can promptly find any information you need. This will also familiarize you with system components and their relationship to one another. After installation and before you apply your own application, check all system functions and features to insure you have installed your VECTOR**STAR** properly.

Proper installation can prevent potential difficulties before they cause harm to your system. Be sure to follow all instructions carefully and pay special attention to safety.

CONVENTIONS

To assist you in understanding the material in this manual, conventions have been established to enhance reader comprehension. Explanations of these conventions are as follows:

- Safety warnings, cautions, and notes present material that is important to user safety. Be sure to read any safety notices you see as they could prevent equipment damage, personal injury, or even death to you or a co-worker.
- **Bold** text highlights other important information that is critical to system operations.
- CAPITALIZED text stresses attention to the details of the procedure.
- Underlined text emphasizes crucial words in sentences that could be misunderstood if the word is not recognized.
- ***DOUBLE BLOCKED***

  text defines words that are to be typed into the computer by the user to interface with the VECTOR**STAR** system.

- **SINGLE BLOCKED**

  text defines words that are displayed by the VECTOR**STAR** on the computer terminal to inform the user of system operations or problems.

ABBREVIATIONS

CCW	Counter Clockwise
CW	Clockwise
D/L	Direction Limit
GC	Goldline Cable
GCS	Goldline Cable Set
LED	Light Emitting Diode
NEC	National Electrical Code
P/N	Part Number
R/D	Resolver-to-Digital
Regen	Regeneration
TL	Test Limits
UL	Underwriters Laboratories

TABLE OF CONTENTS

CHAPTER 1
SYSTEM DESCRIPTION

1.1 INTRODUCTION

The information in this chapter will enable you to understand VECTOR**STAR**'s basic functions and features. These concepts will allow you to apply them to your own unique applications.

1.2 PRODUCT DESCRIPTION

VECTOR**STAR** is a full-featured, high-performance, AC induction spindle and positioning servo in one compact enclosure¾ it is a totally-integrated package available to motion control users. VECTOR**STAR** combines a positioner, a servo amplifier, and an I/O interface into one unit. VECTOR**STAR** sets new standards for motion control with its simple BASIC-like command structure and sophisticated decision-making capability. VECTOR**STAR** provides the outstanding servo performance that you have come to expect from Kollmorgen. Using a high-performance microprocessor, VECTOR**STAR** does not compromise on either positioner software or servo performance. This single microprocessor closes all servo loops, resulting in a truly integrated positioning system. VECTOR**STAR** has the features and performance you need in your next spindle or positioning application.

1.3 FEATURES

VECTOR**STAR** offers a wide feature set to accommodate real world requirements:

- LOW COST

VECTOR**STAR** is very affordable—even though it is full of advanced features. Use all or only a portion of these features to accomplish your application.

- EASY INSTALLATION

VECTOR**STAR** is easy to install because the servo amplifier and the positioner are integrated into one package. Many interconnects, including the tachometer and encoder, are eliminated.

- SIMPLE PROGRAMMING LANGUAGE

VECTOR**STAR** uses simple BASIC-like commands such as RUN, GOTO (for branching), and GOSUB / RETURN (for subroutines). In addition to a simple comparison statement, advanced IF / ELIF / ELSE / END IF statements result in more readable and less error-prone programs. You can comment every line in your program.

- ADVANCED MOTION CONTROL MOVES

The simple language does not prevent you from solving complex problems. VECTOR**STAR** has separate acceleration and deceleration rates, as well as linear, half S-curve,

and full S-curve acceleration profiles. VECTOR**STAR** has Macro Moves for applications where simple indexes cannot do the job. A Macro Move is a combination of up to 30 accelerations, traverses, and decelerations, which are fully precalculated for faster execution. You can program teach modes where position end points can be changed by a factory operator

· MASTER/SLAVE - ELECTRONIC GEARBOX

The electronic gearbox is used to link two motors together so that the velocity of the slave is proportional to the velocity of the master. The ratio can be from 32767:1 to 1:32767 and can be negative to allow the slave to move in the opposite direction. Also, the "index-on-gearing" feature permits phase adjustments.

· MASTER/ SLAVE - PROFILE REGULATION

With profile regulation you can control the slave's motion profile according to an external master motor or frequency. Profile regulation modifies the velocity and acceleration of the slave axis without affecting the final position of the move. You can use profile regulation to implement "feed rate override."

· MOTION GATING AND REGISTRATION

VECTOR**STAR** can precalculate moves to begin motion within one millisecond after a transition on the GATE input. This provides rapid and repeatable motion initiation. VECTOR**STAR** has the ability to capture the current position within 25 microseconds after a transition of the HOME input. This results in fast homing and accurate registration sequences.

· MATHEMATICS Algebraic math is provided for commands such as:

$$X1 = 2 \times (X2 + X3)$$

VECTOR**STAR** has 100 program labels, 50 user-definable variables, and 50 user-definable switches. It also has 15 mathematical/logical operations and over 150 system variables.

· USER UNITS

Quantities such as position, velocity, and acceleration are automatically scaled into user-defined units. This feature lets you program your VECTOR**STAR** in convenient units, such as feet, inches, miles, RPM, and degrees.

· SUPERIOR SERVO LOOP CONTROL

VECTOR**STAR** offers smooth, high-resolution control. Standard VECTOR**STAR** position repeatability is better than one arc-minute, bidirectional. VECTOR**STAR** has a 32-bit position word. Its position loop completely eliminates the digital dither normally associated with positioning systems. Long-term speed stability is 0.01%. The standard system converter (12-bit) provides a resolution of 0.0005 RPM and a maximum speed of 12,000 RPM.

· SELF-TUNING

VECTOR**STAR** can tune itself. You do not have to be a servo expert to set up a system quickly. Just specify the desired bandwidth, and let VECTOR**STAR** do the rest.

· POWERFUL MICROPROCESSOR

The heart of the VECTOR**STAR** system is the 16-bit processor that delivers high performance. The result: VECTOR**STAR** can control a motor and execute its motion program faster than a standard positioner can.

· DIGITAL SERVO LOOPS

Both the position and velocity loops are totally digital. The digital loops give VECTOR**STAR** features not available in standard velocity drives, such as self-tuning, very low velocity offset, and digitally-adjustable servo tuning parameters. The standard analog input permits you to use VECTOR**STAR** as an analog velocity drive.

· FEED-FORWARD GAIN

The digital feed-forward gain reduces following error and motion initiation delay, thereby increasing machine throughput.

· DIAGNOSTICS

VECTOR**STAR** offers a complete set of error diagnostics. When an error occurs, VECTOR**STAR** displays an English language error message. VECTOR**STAR** remembers the last 20 errors even through power loss. In addition, VECTOR**STAR** lets you write your own error handler. During a fault condition, you can use the error handler to set outputs, alert an operator, and shut down your process smoothly. VECTOR**STAR** offers trace and single-step modes so that you can debug your program. VECTOR**STAR** has complete fault monitoring, including travel limit switches, feedback loss, and software position limits, as well as hardware safety circuits (watchdogs) and checksums for more reliable and safer operation.

· I/O

VECTOR**STAR** has up to 32 I/O sections that you connect via ribbon cable to standard OPTO-22 compatible I/O boards or to our own I/O-32. The I/O-32 provides either fixed 24-volt or removable, industry standard, optically-isolated I/O.

· SERIAL COMMUNICATIONS

VECTOR**STAR**'s serial communications provide a powerful link to other popular factory automation devices such as PLC's, process control computers, and smart terminals. VECTOR**STAR** offers RS-232 for most terminals and RS-422/RS-485 for multidrop communications. With multidrop you can put up to 26 axes on one serial line. VECTOR**STAR** can autobaud from 300 baud to 19.2k baud, eliminating the need to set dip switches to start communicating.

· MOTIONLINK

Kollmorgen also offers MotionLink Plus, a powerful, windows-based communications package for your IBM-PC (c) compatible computer. With this package, VECTOR**STAR**'s programs and variables can be retrieved from or saved to a disk drive. Also, on-line help and a full screen editor are built into MOTION LINK.

· MENU-DRIVEN SOFTWARE

VECTOR**STAR**'s programming language allows you to write operator-friendly, menu-driven software. By incorporating an Kollmorgen Data Entry Panel, or any other serial communications device, the operator can be prompted for specific process data.

· MONITOR MODE

VECTOR**STAR** provides interactive communications and permits all system variables and parameters to be examined and modified at any time—even during actual program execution or while the motor is running.

1.4 PART NUMBER DESCRIPTIONS

A model number is printed on a gold and black tag on the front of your VECTOR**STAR**, PA Series, Compensation Card, and External Regen Resistor modules. The model number identifies how the equipment is configured. Each component is described to explain what the model configurations are. You should verify that the model numbers represent the equipment desired for your application. Also verify the compatibility between components of the servo system. The model numbers are as follows:

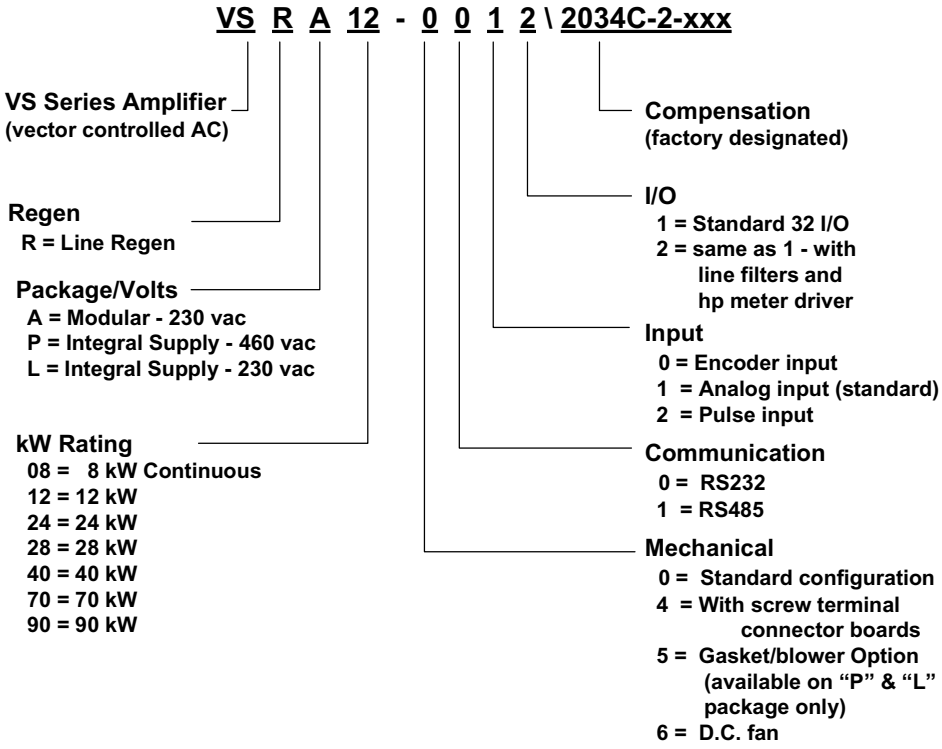


Figure 1.1 VECTORSTAR Model Number Scheme

Table 1.1 VECTORSTAR Description, Low Power

Assembly	Description
VSA08-0012	30 AMP, Replaces VFS5-230
VSA12-0012	40 AMP, Replaces VFS5-240
VSA24-0012	75 AMP, Replaces VFS5-275
VSA28-0012	85 AMP, Replaces VFS5-285
PA8500	ServoStar 85 AMP P.S.

Table 1.2 VECTORSTAR Description, High Power

Assembly	Description
VSP40-0012	Resist. Regen., 460V, 40 HP
VSP70-0012	Resist. Regen., 460V, 75 HP
VSP90-0012	Resist. Regen., 460V, 100 HP
VSR40-0012	Line Regen., 460V, 40 HP
VSR70-0012	Line Regen., 460V, 75 HP
VSR90-0012	Line Regen., 460V, 100 HP
VSL40-0012	Resist. Regen., 230V, 40 HP
VSRL40-0012	Line Regen., 230V, 40 HP
VSL25-0012	Resist. Regen., 230V, 30 HP
VSP25-0012	Resist. Regen., 460V, 30 HP
VSRL25-0012	Line Regen., 230V, 30 HP
VSRP25-0012	Line Regen., 460V, 30 HP

A partial model number is printed on a gold and black tag on the front of the compensation module (the gray plastic box secured by two screws to the front of your VECTOR**STAR**). See Figure 1.2 for the descriptions of the model number (that is, what ABB and HHHH mean). The model number is as follows.

The compensation module depends on your motor and the voltage and current rating of your VECTOR**STAR**. It is important that the motor, the VECTOR**STAR**, and the compensation module model numbers all agree. For example, if your VECTOR**STAR** model number is

VSA24-0012/2604C-011,**

then your compensation module model number must be:

2604C,

and your motor must be a V-2604C. An example of a 2604C motor model number is:

V-2604-CN22.

CAUTION

YOU MUST HAVE THE PROPER COMPENSATION MODULE INSTALLED FOR YOUR MOTOR AND VECTORSTAR. THE COMPENSATION MODULE CHANGES IF THE AMPLIFIER RATINGS CHANGE, EVEN FOR THE SAME MOTOR.

Failure to install the proper compensation module can cause damage to the VECTORSTAR, the motor, or both.

NOTE

Contact Kollmorgen Application Engineering to size regeneration capability.

1.4.5 Molex Assembly Tools

VECTOR**STAR** series electronics use Molex MINI-FIT JR. series connectors. The necessary connectors and pins are included in your VECTOR**STAR** and PA 50/85 connector kits.

You can obtain the crimping and extraction tools from your nearest Molex distributor or by contacting Molex at (708) 969-4550.

Hand Crimping Tool	Molex Order# 11-01-0122
Extractor Tool	Molex Order# 11-03-0038

Power supplies have Phoenix screw-type connectors.

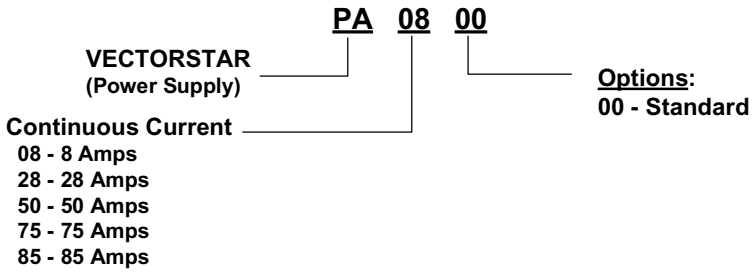


Figure 1.2. PA Power Supply Model Number Scheme

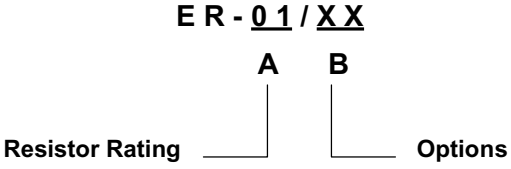


Figure 1.3. External Regen Resistor Model Number Scheme

Table 1.3 External Regen Resistor Model Number Scheme

LEGEND	DEFINITIONS
A	Resistor Rating
21	4.4 Ohms, 1000 W., 230V, 50 & 75 Amp Models Only – PA 50/75
22	2.2 Ohms, 1000 W., 230V, 75 Amp Models Only - PA 75, PA 85
23	2.2 Ohms, 2000 W., 230V, 75 Amp Models Only – PA 75, PA 85
B	Options
00	None available at this printing.

Table 1.4. Environmental Specifications

Operating Temperature*: All units fan cooled	0° C to 45° C
Storage Temperature	-20° C to 70° C
Humidity (Non-Condensing)	10% to 90%

* For operation ambients above 45ºC, consult the Applications Group at Industrial Drives.

1.5 THEORY OF OPERATION

Figure 1.5 shows a system overview.

· MICROPROCESSOR SYSTEM

The VECTOR**STAR** is a digital positioner and servo drive combined into one unit. The velocity loop is 100% digital. The VECTOR**STAR** has battery backup RAM to remember your program and most variables through power-down.

· RESOLVER-TO-DIGITAL CONVERTER

The VECTOR**STAR** is based on a Resolver-to-Digital (R/D) converter. The R/D generates a tachometer signal for your convenience. However, the VECTOR**STAR** does not use the analog tach signal.

· SERIAL PORT

The VECTOR**STAR** has a serial port for communications. This port allows you to monitor the operation, issue commands, and transmit a program.

· DISCRETE INPUTS

The VECTOR**STAR** has 23 discrete inputs, including REMOTE ENABLE, which is on Connector C2 only. Note that two signals, HOME and CYCLE, can be input to the VECTOR**STAR** on two connectors, C2 and C7. Connector C2 provides these three signals with optical isolation. Connector C7 expects non-isolated TTL signals on a 26-pin ribbon cable connector. Optional Connector C8 expects non-isolated TTL signals on a 50-pin ribbon cable connector.

· DISCRETE OUTPUTS

The VECTOR**STAR** has 10 discrete outputs. Notice that O1 appears both on Connector C2 with optical isolation and on Connector C8.

· ENCODER INPUT

The VECTOR**STAR** accepts external inputs in encoder format. This can come from a master motor in a master/slave system. Note that you must use a resolver, even if you use a feedback encoder with the VECTOR**STAR**.

· ENCODER EQUIVALENT OUTPUT

The VECTOR**STAR** provides encoder format output derived from the R/D converter.

· ANALOG INPUT

The VECTOR**STAR** can accept a ±10 volt analog input. This input is converted to digital format by the VECTOR**STAR**. Gain and offset adjustments are made digitally inside the VECTOR**STAR**, not with potentiometers.

· PULSE INPUT (OPT2 CARD)

The VECTOR**STAR** can accept special pulse inputs. The standard VECTOR**STAR** can accept signals directly from encoders or encoder-like devices. As an option, the VECTOR**STAR** can accept other pulse formats, such as count/direction or up/down.

· LED'S

The VECTOR**STAR** provides LED's for diagnostics. These LED's are on the front panel of the VECTOR**STAR**. The LED's are listed below:

ACTIVE
SYS OK
CPU
FAULT
RELAY

· CURRENT LOOP COMPENSATION

The VECTOR**STAR** has analog current loops. The current loop compensation components are all contained in the compensation module located on the front of the VECTOR**STAR**. The current loop compensation changes when you change the motor model. You must install the correct compensation module when changing motor models.

YOU MUST HAVE THE PROPER COMPENSATION MODULE INSTALLED FOR YOUR MOTOR

Failure to install the proper compensation module can cause damage to the VECTORSTAR, the motor, or both.

1.6 VECTOR CONTROL OF INDUCTION MOTORS

Included in the VECTOR**STAR** series is a line of "vector controlled" induction motor drives for spindle applications. This line of drives is also user programmable to fit many applications. The motors used with these drives are designed for machine tool duty and have bearings and special balance for high speed operation.

Operation of the VSA Spindle Drive is identical to that of the standard VECTOR**STAR** system. The control software is designed to control induction motors such as machine tool spindle applications.

The VSA units have an efficient, bonded fin heatsink; a fan shroud that channels all the air from the fan over the heatsink; and an externally mounted fan. This allows the unit to have a contiuous duty rating of 75/85 AMPS.

All VSA units use an EXT1 I/O card, which has filtered channels for the external inputs and a built-in power meter driver circuit. This circuit drives an external analog meter that is calibrated in percentage of power at the motor shaft. Full scale is 10 VDC, which corresponds to 150% power. All 16 inputs and 8 outputs from this EXT1 I/O card can be used to receive and send logical signals to the higher level controllers such as CNCs. These I/O lines can be defined by the user to implement all spindle or other functions with a higher controller.

As an option, a set of user programs can be embedded in the VECTOR**STAR** software, which can perform spindle and customized functions through the I/O lines after power-up. A software switch is used to disable the embedded programs so that user programs can be executed.

All VECTOR**STAR** drives accept the standard Motion Link communications package, which can be accessed via an IBM-type PC or the Kollmorgen Data Entry Panel.

The VECTOR**STAR** drive can also be used as a brushless motor control if the VECTOR**STAR** firmware is installed and the correct compensation is entered into the drive.

An externally mounted Regen Resistor is part of this system. The power meter connects to the 50-pin connector on the EXT1 I/O card. A calibration procedure for the load meter is available from Kollmorgen (the drives are calibrated prior to shipping).

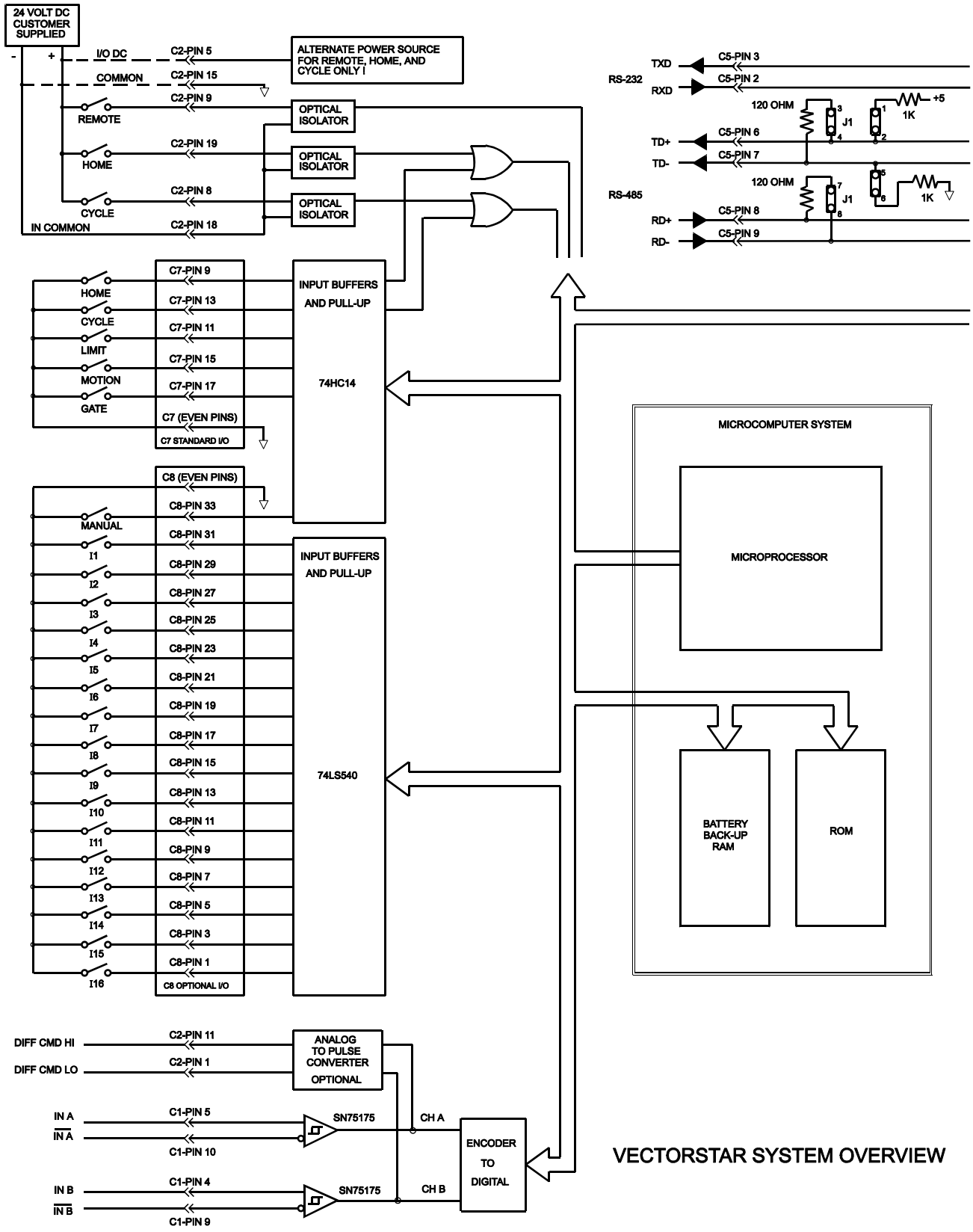
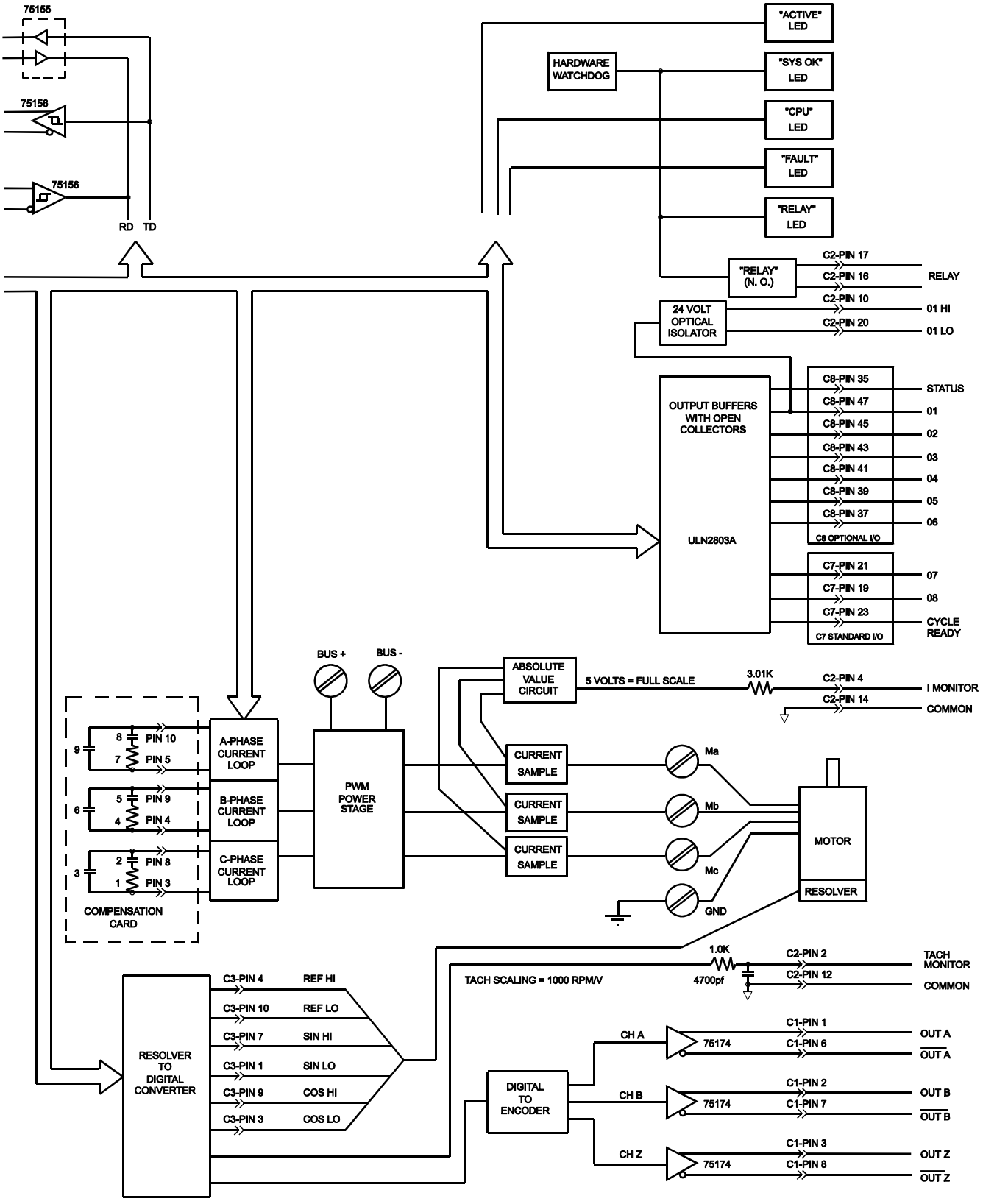


Figure 1.4 VECTORSTAR System Overview


75155
75156
75156
RD
TD
"ACTIVE" LED
HARDWARE WATCHDOG
"SYS OK" LED
"CPU" LED
"FAULT" LED
"RELAY" LED
"RELAY" (N. O.)
C2-PIN 17
C2-PIN 16
RELAY
24 VOLT OPTICAL ISOLATOR
C2-PIN 10
01 HI
C2-PIN 20
01 LO
OUTPUT BUFFERS WITH OPEN COLLECTORS
ULN2803A
C8-PIN 35
STATUS
C8-PIN 47
01
C8-PIN 45
02
C8-PIN 43
03
C8-PIN 41
04
C8-PIN 39
05
C8-PIN 37
06
C8 OPTIONAL I/O
C7-PIN 21
07
C7-PIN 19
08
C7-PIN 23
CYCLE READY
C7 STANDARD I/O
BUS +
BUS -
ABSOLUTE VALUE CIRCUIT
5 VOLTS = FULL SCALE
3.01K
C2-PIN 4
I MONITOR
C2-PIN 14
COMMON
9
8
PIN 10
7
PIN 5
6
5
PIN 9
4
PIN 4
3
2
PIN 8
1
PIN 3
COMPENSATION CARD
A-PHASE CURRENT LOOP
B-PHASE CURRENT LOOP
C-PHASE CURRENT LOOP
PWM POWER STAGE
CURRENT SAMPLE
CURRENT SAMPLE
CURRENT SAMPLE
Ma
Mb
Mc
GND
MOTOR
RESOLVER
1.0K
4700pf
C2-PIN 2
C2-PIN 12
TACH MONITOR
COMMON
TACH SCALING = 1000 RPM/V
RESOLVER TO DIGITAL CONVERTER
C3-PIN 4
REF HI
C3-PIN 10
REF LO
C3-PIN 7
SIN HI
C3-PIN 1
SIN LO
C3-PIN 9
COS HI
C3-PIN 3
COS LO
DIGITAL TO ENCODER
CH A
75174
C1-PIN 1
C1-PIN 6
OUT A
OUT A
CH B
75174
C1-PIN 2
C1-PIN 7
OUT B
OUT B
CH Z
75174
C1-PIN 3
C1-PIN 8
OUT Z
OUT Z

CHAPTER 2
INSTALLATION

2.1 INTRODUCTION

The information in this chapter will familiarize you with safety information, unpacking and inspection, installation requirements, mounting procedures and wiring for installing the VECTOR**STAR**, PA unit, and/or External Regen Resistors. Read the entire chapter carefully. The chapter contains an extensive checkout procedure because most installation problems are caused by incorrect wiring or poor wiring practices. Complete the entire checkout procedure before attempting to put your VECTOR**STAR** system into operation.

2.2 SAFETY INFORMATION

The purpose of this section is to alert you to possible safety hazards associated with this equipment and the precautions you can take to reduce the risk of personal injury and damage to the equipment. Safety notices in this manual provide important information. Read and be familiar with these instructions before attempting installation, operation, or maintenance. Failure to observe these precautions could result in serious bodily injury, damage to the equipment, or operational difficulty.

Figure 2.1. Safety-Alert Symbols

The safety-alert symbols are illustrated in Figure 2.1. When you see these symbols in this manual, be alert to the potential for personal injury. Follow the recommended precautions and safe operating practices included with the alert symbols.

WARNING refers to personal safety. They alert you to potential danger or harm. Failure to follow warning notices could result in personal injury or death.

CAUTION directs attention to general precautions, which if not followed, could result in personal injury and/or equipment damage.

NOTE highlights information critical to your understanding or use of these products.

2.3 UNPACKING AND INSPECTION

Electronic components in this amplifier are static sensitive. Use proper procedures when handling component boards.

Upon receipt of the equipment, closely inspect components to ensure that no damage has occurred in shipment. If damage is detected, notify the carrier immediately.

Carefully remove packing material and remove the equipment from the shipping container. Do not dispose of shipping materials until the packing list has been checked. Parts that are contained within the shipment, but not physically attached to the equipment, should be verified against the packing list. If any parts are missing, notify Kollmorgen at once.

2.4 INSTALLATION REQUIREMENTS

Proper installation and field wiring are of prime importance when considering the application of servo amplifiers. Many problems may be avoided if installation of the equipment is done properly. Users should familiarize themselves with and follow installation and wiring instruction in addition to all applicable codes, laws, and standards. Pay special attention to the following topics when installing Kollmorgen equipment.

2.4.1 Environmental Considerations

The environment in which this equipment is placed can dramatically affect its operation. Kollmorgen recommends that the VECTOR**STAR** and PA unit be operated and stored under the following conditions:

- Operating Temperature: 0° C to 45° C
- Storage Temperature: -20° C to 70° C
- Humidity: 10% to 90% (Non Condensing)

2.4.2 Enclosures

It is suggested that the VECTOR**STAR** and PA unit be mounted in a cabinet or other suitable enclosure to protect them from physical and environmental damage. Refer to specifications for complete system dimensions.

Allow sufficient clearance for the large "regenerative" heat producing resistor(s) mounted at the upper edge of the PA unit and the externally mounted regen (shunt regulator) power resistor(s).

2.5 MOUNTING

The VECTOR**STAR** VSA Series and PA Series power supply should be mounted in a cabinet or other suitable enclosure to protect them from physical and environmental damage.

The PA power supply and the VECTOR**STAR** are outfitted with protective guards over the power terminal blocks. After wiring is complete, always replace the protective guards to reduce shock hazard.

REPLACE THE POWER TERMINAL GUARDS AFTER WIRING THE SYSTEM!

2.5.1 Mounting the VECTORSTAR

Refer to Figure B10 in Appendix B for mounting dimensions. The VECTOR**STAR** and PA should be mounted in the vertical position. Since these units are convection cooled, there should be a minimum of 25 millimeters (1 inch) of unobstructed space above and below the units. In addition, there should be a minimum of 20 millimeters (0.8 inch) between units to ensure proper airflow for these convection-cooled units.

2.5.2 Mounting the PA

The PA Power Supply module may be mounted on either side of the VECTOR**STAR**. However, a 20 millimeter (0.8 inch) space is required between units.

Some PA units have resistors that dissipate the energy returned to the PA during regenerative braking. These are referred to as regen resistors. Some applications require externally mounted regen resistors, while others may require none at all. To determine if your PA has internal regen resistors, check its model number.

REGEN RESISTORS GET HOT!

If your PA has internal regen resistors, allow sufficient clearance above the PA .

Refer to Figures B.1-B.8 in Appendix B for more information concerning mounting.

Refer to appropriate outline and dimension drawings in Appendix B for more information:

Figure B.10	VECTOR**STAR** 24 and 28 Amp Unit
Figure B.9	PA 50, 75 and 85 Amp Unit

2.5.3 Mounting the External Regen Resistor

External mounting of regen resistors is an option on PA units. To determine if your unit uses external regen resistors, check its model number. Also, the model number indicates the standard resistor value. These resistors should be enclosed to reduce shock hazard. Regen resistors get hot! They are a burn hazard and they are a fire hazard. They can produce enough heat to melt insulation. Enclose these resistors! The enclosure must provide ventilation and withstand high temperatures. Do not mount the resistors underneath the VECTOR**STAR** or PA .

EXTERNAL REGEN RESISTORS ARE A SHOCK HAZARD!

Mount these resistors properly! Enclose these resistors to protect personnel and equipment!

EXTERNAL REGEN RESISTORS GET HOT!

If you are using external regen resistors, allow sufficient clearance around the resistors. Enclosures must provide ventilation.

2.6 WIRING

The customer is responsible for providing proper circuit breaker or fuse protection. The customer is responsible for providing proper wire gauge and insulation rating for all wiring, including motor, AC line, DC bus, and External Regen Resistor connections. The customer is responsible for making sure that all system wiring and electrical protection comply with all applicable national and local electric codes.

Figures B.1-B.8 in Appendix B and Figure 3.1 in Chapter 3 illustrate the System Wiring Diagram. Carefully read all 9 figures before wiring your VECTOR**STAR** system, including all of the notes following Figure B.8.

Before wiring your VECTOR-STAR system, carefully study all eight pages of Figure B.1-B.8 in Appendix B, including ALL of the notes following Figure B.8, and Figure 3.1.

When wiring your VECTOR**STAR** system, observe the following guidelines:

1. Twist all AC leads to minimize electromagnetic emissions (noise).

2. Avoid running signal leads in close proximity to power leads, motor stator leads, or other sources of electromagnetic noise. Run signal leads in separate conduit from power leads. Shields are recommended for signal leads.

2. Minimize lead lengths.

4. Connect the VECTOR**STAR** system according to the System Wiring Diagram, Figures B.1-B.8; pay close attention to the grounding scheme and notes.

2.6.1 Wiring the Ground

Two ground terminals are located on the front of the VECTOR**STAR** and PA. These ground screws are directly below the Power Terminal Block of each unit.

To prevent shock hazard and to ensure proper operation of the system, the VECTOR**STAR**, PA, and the motor must be grounded.

GROUND YOUR VECTORSTAR, PA , and motor PROPERLY! Failure to provide proper grounding can result in a shock hazard.

One of the ground screws on the PA should be connected to Earth Ground or Machine Ground. The other should be connected to the adjacent VECTOR**STAR** ground screw, along with the ground wire from the motor.

The other ground screw on the VECTOR**STAR** chassis should then be connected to the next VECTOR**STAR**, etc. Use 10 AWG or larger wire for grounding between VECTOR**STAR**s and PAs and from PA to Earth Ground or Machine Ground.

2.6.2 Wiring the Power Connections

This section discusses how to wire the PA and VECTOR**STAR** power connections.

To prevent equipment damage, the AC Line and the DC bus must be connected as indicated by the System Wiring Diagram, Figures B.1-B.8.

The PA and VECTOR**STAR** are configured at the factory for operation from a 230 volt AC Line. The PA and the VECTOR**STAR** must both have the same voltage rating. Operating a PA or a VECTOR**STAR** with incorrect voltage can damage the units.

Incorrect motor wiring can cause erratic or runaway motor operation. Use of factory supplied cable sets is highly recommended.

2.6.2.1 Wiring the Motor

Connect Pins A, B, and C of the motor connector through a motor cable to Ma, Mb, and Mc, respectively, on the Power Terminal Block located on the front of the VECTOR**STAR** amplifier. Connect Pin D of the motor connector through the same motor cable to the VECTOR**STAR** chassis ground screw located directly beneath the Power Terminal Block.

It is very important that you wire the motor properly. Brushless permanent magnet motors are not like induction motors. You cannot simply interchange two phases to reverse the direction of rotation.

Table 2.1. Motor Cable Color Code

Motor Connector	Wire Marker	VECTOR**STAR** Power Terminal
Pin A	M_a	Terminal M_a
Pin B	M_b	Terminal M_b
Pin C	M_c	Terminal M_c

Incorrect motor wiring can cause erratic or runaway motor operation. Use of factory supplied cable sets is highly recommended.

2.6.2.2 Motor Protection

Under normal conditions, you do not need to add protection for your motor; the VECTOR**STAR** is normally configured to limit the continuous current below the rating of the motor. However, if the drive is oversized (that is, it can provide continuous current that greatly exceeds the motor's continuous current rating), you may want to add motor overload relays in series with the motor. Also, you can wire a contactor in series with the motor leads. **Always be careful to wire the motor properly.**

2.6.2.3 Motor Thermostat

Kollmorgen V-Series motors have a thermostat. The thermostat contacts are closed during normal operation and open when the motor overheats. The VECTOR**STAR** does not provide a direct input for the motor thermostat. You should connect the thermostat contacts to a VECTOR**STAR** general purpose input, and program your VECTOR**STAR** to bring about an orderly shutdown when the thermostat opens. The thermostat ratings are:

Rated Voltage: 277 VAC at 60 Hz
250 VAC at 50 Hz

Rated Current: 2.5 Amps at power factor of 1.0
1.6 Amps at power factor of 0.6

Resistance: 50 m Ohms

The thermostat resets (closes) when the motor cools. The customer is responsible for making sure that the

motor does not begin moving unexpectedly when the thermostat closes. Do not connect the thermostat directly in line with the VECTORSTAR REMOTE ENABLE or LIMIT inputs, as this may allow the system to begin operation unexpectedly. You must latch the thermostat switch, either in hardware or software. It can take several minutes for the motor to cool enough to allow the thermostat to close. Unexpected operation several minutes after a fault is a safety hazard.

LATCH THE MOTOR THERMOSTAT!

DO NOT CONNECT THE THERMOSTAT DIRECTLY TO THE VECTORSTAR REMOTE ENABLE OR LIMIT INPUTS.


WARNING

The motor thermostat resets (closes) when the motor cools. This can allow the motor to restart operation unexpectedly after a motor thermostat fault unless the thermostat is latched. Always latch the thermostat.

Some electrical noise from the motor leads will couple capacitively with the thermostat leads. This occasionally causes false thermostat trips, which means your controller senses that the thermostat opened even though the thermostat contacts were closed. If this occurs you can use the thermostat contacts to energize a relay and then connect the relay contacts to your controller. The electrical noise does not normally cause relay contacts to open.

2.6.2.4 Wiring the DC Bus

Connect the DC Bus from the PA Power Terminal Block (BUS+, BUS-) to the VECTOR**STAR** Power Terminal Block (BUS+, BUS-). You must observe polarity of DC Bus connections: always connect BUS+ to BUS+ and connect BUS- to BUS. The PA-50 and -75 can be connected to any combination of 4 VECTOR**STAR**s, although no more than 2 units can be on one side of the PA.


WARNING

Failure to observe polarity of the DC Bus can result in personal injury.


CAUTION

Failure to observe the polarity of the DC Bus will result in damage to the PA and VECTORSTAR.

2.6.2.5 Wiring the AC Line

Connect the three-phase AC Line to La, Lb, and Lc on the Power Terminal Block located on the front of the PA unit.

The PA will work with a single-phase AC Line. 220 VAC single-phase input lines may be connected to any two terminals, La, Lb, or Lc. Note that the PA must be derated for single-phase AC Line operation. A 12-Amp PA with a single-phase AC Line can only provide 10 Amps; a 20-Amp PA with a single-phase AC line can only provide 16 Amps.


CAUTION

The PA must be derated when operated from a single-phase AC Line.

2.6.2.6 Wiring the Regen Resistor

If an external regen resistor is used, wire it to the External Regen Resistor Connector on the PA. Note that you must specify that you need an external regen resistor when ordering your PA as this is an option. Refer to Notes 3 and 11 in Appendix B.

2.6.3 Wiring the PA Front Panel Connectors

This section will discuss wiring of the PA front panel connectors. The mating connectors are supplied with the PA.

2.6.3.1 Wiring the Control Power to PA

The Control Power for the PA is 190-260 VAC. Connect the Control Power to Connector "Control Input" of the PA. Connect one AC line to Pin 1 and the other to Pin 3.

2.6.3.2 Wiring to the PA Fault Output on "Bus OK"

The PA Fault Output Contact closes approximately 250 milliseconds after power is applied to the PA. This contact opens if a fault occurs in the PA. This is a relay contact from Pin 1 to Pin 2 of Connector "Bus OK." You can connect this contact to your controller or to a VECTOR**STAR** remote inhibit. You should inhibit the entire system if a PA fault occurs! See Note 2 in Appendix B.

The ratings of this relay are:

2 Amps at 28 Volts DC, resistive.
1 Amp at 120 Volts AC, resistive.

CAUTION

INHIBIT YOUR SYSTEM IF A PA FAULT OCCURS! YOU MUST WIRE YOUR SYSTEM FOR THIS FUNCTION!

2.6.3.3 Wiring PA Connector "Logic"

Connector "Logic" of PA is the Logic Power Supply for the VECTOR**STAR**. Wire the Logic Power Supply from this connector on the PA to Connector C4 of the VECTOR**STAR**. Each logic power supply voltage is connected to two pins so that you can wire from the PA to the nearest VECTOR**STAR**. For example, +15 volts appears side by side on Pins 1 and 5. Note that the PA 12 and 20 Amp models can have a maximum of three units (axis) connected, while the 50 and 75 amp models may have six units connected. Table 2.2 lists the ratings of the PA logic power supplies.

Table 2.2. PA Logic Power

VOLTAGE	CURRENT (PER AXIS)	VECTOR-**STAR** Connector C4 PIN	PA Connector Logic
+ 15 VDC ∂20%	0.25 AMP	1, 5	1
- 15 VDC ∂20%	0.25 AMP	2, 6	2
COMMON	---	3, 7	3
+ 8 VDC ∂20%	1 AMP	4, 8	4
LOGIC POWER SUPPLY MAXIMUM RATINGS			

CAUTION

Failure to observe the polarity of the logic power supply will result in damage to the PA and VECTORSTAR.

2.6.4 Wiring the VECTORSTAR Front Panel Connectors

This section will discuss wiring of the VECTOR**STAR** front panel connectors: C1, C2, C3, C4, C5, C6, and C10. Mating connectors for C1-C8 and C10 are supplied with the VECTOR**STAR**.

2.6.4.1 Wiring C1, Encoder Equivalent

The Encoder Equivalent Connector connects encoder inputs and encoder equivalent outputs. The VECTOR**STAR** uses standard encoder format (A-B quadrature). This format has excellent noise immunity because only one channel changes at a time. As an option, pulse inputs in different formats are also supported.

The Encoder Equivalent Connector is used for master/slave systems. If your VECTOR**STAR** is the slave axis, then connect the output from the master to INA and INB. If your VECTOR**STAR** is the master axis, connect OUTA, OUTB, and OUTZ to the inputs on the slave axis. See Figure 2.2 and/or Figures B.1-B.8 for connection diagrams.

All encoder signals are differential (as opposed to single-ended) to increase noise immunity. This means that each signal is transmitted with its logical inverse (for example, OUTA and OUTA'). Logical inverse means that if OUTA is 5 volts, then OUTA' is 0 volts and that if OUTA is 0 volts, then OUTA' is 5 volts. (Note that Figure B.1-9 shows the inverse of OUTA as OUTA with a bar drawn directly above it; here, the inverse of OUTA will be designated OUTA'.)

The encoder equivalent inputs and outputs conform to RS-485. One standard RS-485 output can drive up to 32 standard RS-485 inputs, provided that the capacitance of the interconnecting cable is small enough. Capacitance increases with cable length, which implies that the transmitter (OUT's) and receivers (IN's) should be as close to each other as is practical.

You should use 120 ohm cable. For longer distances (over a 100 feet), consider using reduced capacitance cables such as those available from Black Box (Pittsburgh, PA). The cable should be run from point to point, as opposed to branching out from a single point. Branches could cause reflections (a transmission line effect) that can interfere with the signals. If the cable is very long, ringing (also a transmission line effect) may occur. If this happens, you should connect a 120 ohm resistor across each signal and its logical inverse at both ends of the cable.

The Encoder Equivalent Connector uses RS-485 compatible 75174 line drivers and 75175 line receivers, which are available from many IC manufacturers, including Texas Instruments.

The VECTOR**STAR** uses the same phasing for the encoder inputs and outputs; for clockwise rotation, Channel A leads Channel B. For standard systems, the

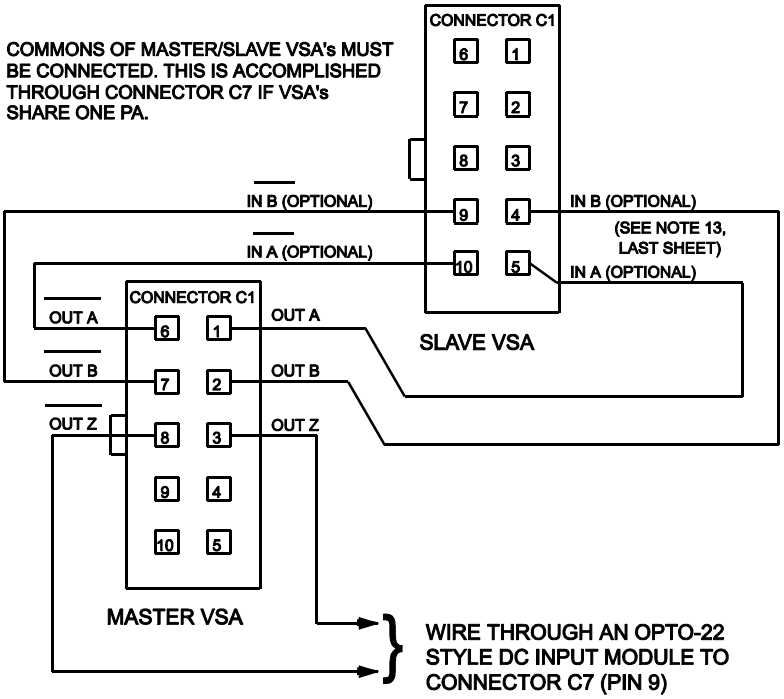


Figure 2.2 Master VECTORSTAR and Slave VECTORSTAR

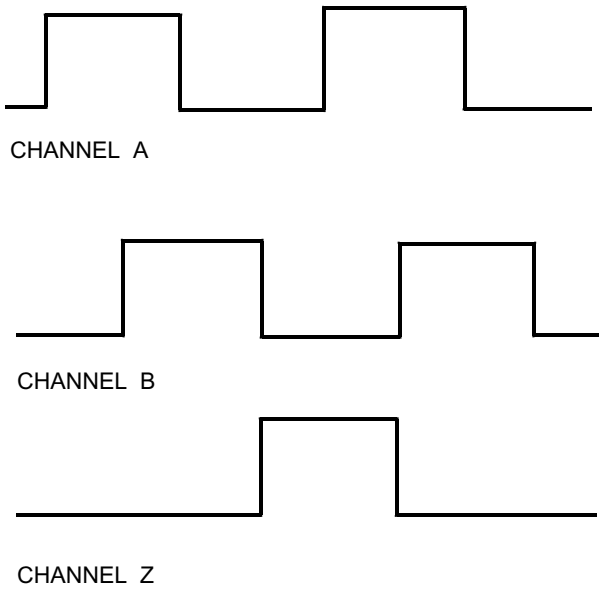


Figure 2.3 Encoder Phasing for Clockwise Rotation

$\overline{\text{OUT A}}$	6	1	OUT A
$\overline{\text{OUT B}}$	7	2	OUT B
$\overline{\text{OUT Z}}$	8	3	OUT Z
$\overline{\text{IN B}}$ (Standard)	9	4	IN B (Standard)
$\overline{\text{IN A}}$ (Standard)	10	5	IN A (Standard)

Figure 2.4. VECTORSTAR (C1)

inputs are Channel A (Pins 5, 10) and Channel B (Pins 4, 9). The outputs are Channel A (Pins 1, 6), Channel B (Pins 2, 7), and the marker channel, Channel Z (Pins 3, 8). There is no input Channel Z, though this signal often would be connected to HOME (see Chapter 8). Refer to Figure 2.3 for a phasing diagram.

The logic power supply common is not available on Connector C1. If you are wiring from one VECTOR**STAR** to another VECTOR**STAR** on the same PA, then you need not connect common, as the commons of the two drives will be connected through the logic power supply connector, C4. If you need to connect to common, you must obtain it from Connector C2 or C4. Be sure that the common of the VECTOR**STAR** and of the encoder power supply are electrically connected to each other.

2.6.4.2 Encoder Equivalent Input

The encoder and pulse inputs are for signals from a master encoder (for example, an encoder on another motor, or the encoder output of another VECTOR**STAR**). The VECTOR**STAR** can also use an encoder for feedback in some cases. The encoder equivalent input is not available when pulse input (OPT2 Card) is installed.

If the encoder input is being used for a velocity command, as is the case for electronic gearbox applications, then you must be careful to connect the encoder input for the correct direction of rotation. If the VECTOR**STAR** is being used as a velocity drive, and another controller is closing a position loop, reversing the encoder inputs can cause the VECTOR**STAR** to run away or oscillate. Be careful to connect the encoder/pulse input so that the direction is correct!

Phase the Encoder Equivalent Input correctly. Incorrect phasing of the encoder input can cause excursions, oscillations, or runaways.

2.6.4.3 Pulse Input (OPT2 Card)

As an option, you can also purchase your VECTOR**STAR** with a Pulse Input option. This option allows you to:

1. Control the VECTOR**STAR** from a stepper-motor controller. For this function, configure the VECTOR**STAR** as an electronic gearbox and select the format that is compatible with your controller.
2. Input a single-phase clock from your computer. There are occasions when you want to control the VECTOR**STAR** with a unidirectional clock (for example, feedrate override). The Pulse Input option accepts a single-phase clock, such as would be generated from an electronic timer/counter chip.

The Pulse Input Option board (VECTOR**STAR**-OPT2) is mounted inside the VECTOR**STAR** unit. This board accepts pulse inputs in three formats: count/direction, up/down, or single-phase. Use the toggle switch on the board to select the formats.

2.6.4.4 Encoder Equivalent Output

The Encoder Equivalent Output provides position information to another device in the system. The output is in standard encoder format. The Encoder Equivalent Output must be phased correctly. If the VECTOR**STAR** is being used as a velocity drive, and another controller is closing a position loop, reversing the Encoder Equivalent Output can cause the VECTOR**STAR** to run away or oscillate. Be careful to connect the encoder equivalent output so that the direction is correct!

Phase the Encoder Equivalent Output correctly. Incorrect phasing of the Encoder Equivalent Output can cause excursions, oscillations, or runaways.

2.6.4.5 Wiring C2 and C1O, Customer I/O and Analog Input

The VECTOR**STAR** accepts an analog input. If you use the analog input, then the encoder input is not available. If you use the analog input, refer to Chapter 10 for a program that configures the VECTOR**STAR** as a velocity drive.

·	C1O	Analog Input H	Pin 5
		Analog Input L	Pin 6
		Shield	Pin 8

The input is scaled for ±10 volts = full scale. Input impedance is > 20 k Ohm. The analog input is differential; that is, the input voltage is taken to be the voltage at Pin 5 minus the voltage at Pin 6. Differential (two-wire) signals have higher noise immunity than single-wire signals, since noise picked up by the wires is eliminated when the two voltages are subtracted.

Typically, one of the two wires will be at a voltage nominally the same as Common. Preferably, this wire should be connected to Pin 6. It is important to note that swapping Pins 5 and 6 reverses the polarity of this input.

When using the analog input you must be careful to shield it properly. You should use shielded, twisted-pair cable. Connect the shield to Pin 8 on Connector C10 on the VECTOR**STAR**, or to the frame at the source of the signal, or at both ends. Determining the best shield connection is often a matter of trial-and-error. Begin by wiring the shield only at the VECTOR**STAR**.

If the VECTOR**STAR** is being used as an analog velocity drive, you must be careful to connect the analog input with the correct polarity, since the polarity determines the direction of rotation. If the VECTOR**STAR** is being used as a velocity drive, and a different controller is closing a position loop, reversing the polarity of the analog input can cause the VECTOR**STAR** to run away or oscillate. Be careful to connect the analog input so that its polarity is correct!

Phase the analog input correctly. Incorrect polarity of the analog input can cause excursions, oscillations, or runaways.

Connector C2 has a variety of signals:

- Tach Monitor Pin 2

Tach monitor indicates velocity. It is referenced to Common and scaled for 1000 RPM clockwise = +1 volt. This output has a 1k Ohm resistor in series and a 4700 pF filter capacitor connected to Common.

- I Monitor Pin 4

I monitor is a Common-referenced signal scaled for full (peak) current = 5 volts. This output has a 3.01k Ohm resistor in series. I Monitor is always positive.

- Common Pins 12, 14, 15

Logic Power Supply Common is provided on several pins since it is used in several places. Do not connect shields to Common.

- Shield Pins 3, 13

The shield pins are connected directly to the VECTOR**STAR** frame. You should connect shields from cables to the shield pins.

- Relay Pins 16, 17

These two pins are connected to a set of normally open relay contacts. The contacts are open on power-up and closed following the power-up self-tests and autobauding. The relay is controlled by the watchdog timer. If the contacts open once the VECTOR**STAR** is running, then there is a serious fault and you cannot depend on the microprocessor to operate properly. Always wire all VECTOR**STAR** relays to disconnect all power in the event of the relay contacts opening. These contacts are rated for a maximum current of 1 Amp and a maximum voltage of 115 volts AC or DC. The maximum switched power is 30 Watts DC or 60 volt-amps AC.

- CYCLE/CYCLE RETURN Pins 8, 7

CYCLE is on Pin 8; CYCLE RETURN is on Pin 7. CYCLE is normally used to **STAR**t a cycle of a user program; it can also be used as a general purpose input. CYCLE is optically isolated. CYCLE should be pulled up to between 12 and 30 volts with reference to CYCLE RETURN (not Common) to turn on, or activate, the CYCLE input. Exceeding 30 volts could damage this input. Observe polarity when connecting CYCLE and CYCLE RETURN.

Note that CYCLE is also available on Connector C7. The CYCLE signal on the Connector C7 is not optically isolated. If you use both CYCLE inputs, then the VECTOR**STAR** will sense that CYCLE is on if the CYCLE signal from either Connector C2 or C7 is on.

- REMOTE/REMOTE RETURN Pins 9, 6

REMOTE is on pin 9; REMOTE RETURN is on pin 6. REMOTE is an input that must be activated for the VECTOR**STAR** to be enabled. REMOTE is optically isolated. It should be pulled up to between 12 and 30 volts with reference to REMOTE RETURN (not common) to activate the REMOTE input. Exceeding 30 volts could damage this input. Observe polarity when connecting REMOTE and REMOTE RETURN.

- HOME/HOME RETURN Pins 19, 18

HOME is on pin 19; HOME RETURN is on pin 18. HOME is intended to be connected to a home limit switch; it can also be used as a general purpose input. HOME is optically isolated. HOME should be pulled up to between 12 and 30 volts with reference to HOME RETURN (not Common) to activate the HOME input. Exceeding 30 volts could damage this input. Observe polarity when connecting HOME and HOME RETURN.

Note that HOME is also available on Connector C7. The HOME signal on Connector C7 is not optically isolated. If you use both HOME inputs, then the VECTOR**STAR** will sense that HOME is on if the HOME signal from either Connector C2 or C7 is on.

- I/O DC Pin 5

I/O DC is provided for occasions when you do not need optical isolation for HOME, CYCLE, and REMOTE ENABLE. An unregulated, low current power supply is provided on I/O DC to power those three inputs. Do NOT use I/O DC as a power supply for anything except these three isolated inputs!

If you want to use I/O DC, then connect the return lines (HOME RETURN, CYCLE RETURN, and/or REMOTE RETURN) to Common. Then you can connect HOME, CYCLE, and/or REMOTE ENABLE to I/O DC (usually through a relay contact) to activate those inputs.

Using VECTORSTAR internal power supply (I/O DC) for any purpose except power for HOME, CYCLE, and REMOTE ENABLE can damage the PA.

- O1 Pins 10, 20

O1 is an optically-isolated output. It is a solid state relay rated for 0.25 Amps and 30 VDC maximum. Only DC voltages, with the more positive voltage on Pin 10, may be applied to this output. You must observe polarity when connecting O1. You should install a 0.25 amp fuse in series with O1, as it is not fused internally.

Note that O1 is also available on Connector C8. The O1 signal on Connector C8 is not optically isolated. You can use both O1 signals at the same time.

2.6.4.6 Wiring C3, Resolver

Install the resolver cable between the motor and the VECTOR**STAR**. Make the connection at the motor resolver connector and Connector C3 on the VECTOR**STAR**.

Connect the resolver leads correctly. Incorrect motor resolver phasing can cause erratic operation, runaway, or damage to the system. Use of Kollmorgen resolver cable sets is highly recommended.

If you are making your own resolver cables, you must obtain the procedure from Kollmorgen to make cables.

2.6.4.7 Wiring C4, Logic Power Supply

Wiring C4 is discussed above with "Wiring PA Connector C2" since these two connectors are connected to each other.

2.6.4.8 Wiring C5, Serial Communications

Connector C5, the Serial Communications Connector, is on the front of the VECTOR**STAR**. It is a 9-pin, D-type, sub-miniature plug connector (DE-9P). The communications cable must have the mating DE-9S connector. See the System Wiring Diagram (Figure B.3). Note that wiring for RS-232 and RS-485 is completely different. Refer to the model number to determine if your unit is RS-232 or RS-485. Be sure that the common of the VECTOR**STAR** and the power supply common of your computer or terminal are electrically connected to each other, whether you are using RS-232 or RS-485. Use the COM pin on the serial connector (Connector C5) for the VECTOR**STAR** common.

- RS-232 vs. RS-485

The VECTOR**STAR** can communicate with a terminal or a computer over a serial data line with EIA RS-232-C, the most common electrical interface for terminals and personal computers. It can also communicate with EIA RS-485, a standard that supports multiple devices on one serial line. RS-485 is an upgrade of RS-422; RS-422 is restricted to one transmitter per serial line; otherwise, the devices will work with RS-485 devices. You should specify the communication standard that you want when you order your VECTOR**STAR**.

When choosing between RS-232 and RS-485, there are a number of factors to take into consideration. RS-232 is much more common than RS-485. Most terminals and computer serial ports are RS-232; however, RS-485 is becoming more widely available on personal computers. RS-485 is differential; RS-232 is single-ended. To illustrate this, RS-232 transmits one output, TxD. Alternatively, RS-485 simultaneously transmits TxD+ and

TxD-, the logical inverse of TxD+. (Logical inverse means that if TxD+ = 1, then TxD- = 0). RS-232 transmitters typically have series-terminated outputs, while RS-485 transmitters do not. All this makes RS-485 less susceptible to noise and able to transmit over much longer distances. On the other hand, RS-232 is easier to use and much less susceptible to transmission line effects. RS-232 is also less expensive. A major advantage of RS-485 is that it allows multidrop communication, where many axes can be connected on one serial line.

Table 2.3. Communication Requirements

To Communicate With Your VECTOR**STAR**, You Will Need:
RS-232 Terminal or IBM-PC Compatible with Communication Software or RS-485 Terminal and Connector for Your Terminal DE-9S Connector (Provided with VECTOR**STAR**) 3-Wire-with-Shield Cable (RS-232)* 5-Wire-with-Shield Cable (RS-485)*

*Use one of the conductors to connect the power supply commons. The shield should not be used for this connection, but should connect to earth ground.

If you have an RS-232 terminal or computer, and you want to use RS-485, either for multidrop or for noise immunity, then you can purchase an RS-232-to-RS-485 converter. For example, Anaheim Automation (Anaheim, CA) produces Model DC 2170 for this purpose.

The RS-232 serial input and outputs connect to 75155 line driver/receivers. The RS-485 serial input and outputs connect to 75176B line driver/receivers. These chips are available from many IC manufacturers, including Texas Instruments.

· LINE TERMINATION

The VECTOR**STAR** provides line termination for RS-485. An RS-485 line may need to be terminated to reduce ringing on long cables. (In this section, Line refers specifically to the RxD+/RxD- pair or the TxD+/TxD- pair.)

A line is terminated by connecting a resistor from RxD+ to RxD- or TxD+ to TxD-. Terminating resistors are provided inside the VECTOR**STAR**. These resistors are connected by installing jumpers across Pins 3 and 4, and across Pins 7 and 8 on J1 on the front of the VECTOR**STAR**. The lines should only be terminated at the end of the communication cable. For example, if you are using several VECTOR**STAR**s in multidrop, then you should install the jumpers only on the units at the ends of the serial cable.

For RS-485, you should install jumpers across Pins 3-4 and 7-8 on J1 on the front of the VECTORSTAR at the end of the communication cable. If the device you are using to communicate with the VECTORSTAR is at one end of the line, you should terminate the line at that end with a 120 ohm resistor.

· DISTANCE

RS-232 generally works well up to 50 feet using standard cable. Special low capacitance cable can extend this range up to 500 feet, although the baud rate may need to be reduced. These cables are available from many companies, including Black Box (Pittsburgh, PA).

RS-485 can be used up to 4000 feet. Be forewarned, however, that using very long cables may be more difficult than just making the connections. Special attention must be paid to AC loading, transmission line effects, noise pickup, and common-mode voltages. It would be wise to have a person who is knowledgeable in these matters to review the cable design beforehand. Again, baud rates may need to be reduced as the length increases, and low capacitance cables may be required.

You can obtain a copy of the specification for RS-232 or RS-485 by contacting:

Electronic Industries Association
Engineering Department
2001 Eye Street, N.W.
Washington, D.C. 20006

· CONNECTING TO A TERMINAL

Connect the pins on the DE-9S connector that came with your VECTOR**STAR** as listed in Table 2.4, which assumes that your terminal has an RS-232 serial port which is configured as Data Terminal Equipment (DTE), the usual configuration. If your terminal is configured as Data Communications Equipment (DCE), then interchange Pins 2 and 3 at one end of the cable.

Note that handshaking signals (CTS, DTR, etc.) are not supported.

Table 2.4. Terminal Wiring

WIRING TO MOST RS-232 TERMINALS				
VECTOR-**STAR** (DE-9) Female			Terminal (DB-25) Male or Female	
1		SHIELD*		1
2	RxD	⇑	TxD	2
3	TxD	⇓	RxD	3
5		COM		7

*Usually, you will only connect the shield at one end. Sometimes the shield should be connected at both ends. This is system dependent and generally found by trial and error.

Table 2.5. PC Wiring

WIRING TO AN IBM-PC COMPATIBLE COMPUTER							
VECTOR**STAR** (DE-9P) Female			IBM-PC (DB-25S) Female		VECTOR**STAR** (DE-9P) Female		IBM-AT (DE-9S) Female
1		SHIELD*		1	1	SHIELD*	FRAME+
2	RxD	<--	TxD	2	2 RxD	<--	TxD 3
3	TxD	-->	RxD	3	3 TxD	-->	RxD 2
5		COM		7	5	COM	5

*Usually, you will only connect the shield at one end. Sometimes, the shield should be connected at both ends. This is system dependent and generally found by trial and error.

+The PC-AT does not have a pin for shield. If you want to hook the shield to the PC-AT end of the cable, you must connect it directly to the frame of the computer.

· CONNECTING TO A COMPUTER

Connect the pins on the DE-9 connector that came with your VECTOR**STAR** as follows:

Table 2.5 assumes your PC has an RS-232 serial port that is configured as Data Terminal Equipment (DTE), the usual configuration. If your port is configured as Data Communications Equipment (DCE), then interchange Pins 2 and 3 at one end of the cable.

Note that handshaking signals (CTS, DTR, etc.) are not supported.

If the PC-compatible computer is AC-line powered (that is, not battery powered), use extreme caution when interconnecting the PC to the VECTOR**STAR** serial port. Both the PC and the VECTOR**STAR** must share the same power supply common. If they do not, the voltage difference between the two commons could damage either or both machines.

Make sure that the computer and VECTORSTAR share the same power supply common. Either your computer or the VECTORSTAR or both can be damaged if the commons are not at the same potential.

If you use a computer, you will need communications software. The recommended communications software for use with the VECTOR**STAR** is Motion Link, a Kollmorgen software package specially designed for the VECTOR**STAR**. Other communications software packages include XTALK, QMODEM, PROCOM, KERMIT, and PC-TALK. See Chapter 7 for information about Motion Link.

2.6.4.9 Wiring C6, Fan Power

Connect the 230 VAC Control Power from the PA (Connector C8) to the fan power connector (Connector C6). As the wiring diagram (Figure B.1) shows, 230 VAC appears side by side on each connector, similar to the logic power supply connector. This allows you to "daisy-chain" the Control Power to each unit that requires it.

Connector C6 is only used on VECTORSTAR amplifiers with continuous ratings of 20 Amps and above.

The VSA 28 is available with 24 VDC fans.

2.6.4.10 Wiring C7, Standard I/O

C7 is a 26-pin ribbon cable connector. It provides non-isolated, 5-volt TTL-level inputs. Open collector outputs are directly interfaced to 8-line OPTO-22 compatible I/O boards. These boards are widely available and use industry standard optical isolation modules, available from several companies, including Potter and Brumfield, Grayhill, Gordos, OPTO-22, and Crydom. This connector is on top of the VECTOR**STAR**.

Note that you must provide a separate power supply when using standard OPTO-22 compatible I/O boards. This supply must provide 5 volts DC to power the I/O modules on the VECTOR**STAR** side of the isolation. The common of this supply will normally be connected to the common of the VECTOR**STAR** through the even numbered pins on Connector C7. **Do not use the VECTORSTAR 5 VDC power supply to power I/O modules!**

Additionally, a second power supply may be needed to provide power for those devices which are isolated from the VECTOR**STAR** by the I/O modules. The common of this supply should not be connected to the VECTOR**STAR** common.

You must provide an additional power supply for the I/O modules.

- HOME Pin 9

HOME is the input for the home limit switch. It is also the registration input. If your application has both a home limit switch and a **registration** input, use HOME for the registration input and connect the home limit switch to a general purpose input. This input can be used as a general purpose input if these primary functions are not required.

HOME is available on two connectors. See the description of HOME on Connector C2 in section 2.6.4.5.

- LIMIT Pin 11

PART OF QUICK START
This input must be turned on for normal operation. Connect the normally closed contacts of the overtravel limit switches in series with LIMIT. Ground to enable

If your application does not allow you to use hardware travel limit switches, then you must hardwire LIMIT on. If you are not using Connector C7 you can hardwire LIMIT on by installing a jumper directly on C7 between Pins 11 and 12. The VECTOR**STAR** is shipped from the factory with this jumper installed.

- CYCLE Pin 13

CYCLE is an input that is normally used to start a cycle of a user program. It can be used as a general purpose input if this primary function is not required. See the description of CYCLE on Connector C2 above.

- MOTION Pin 15

PART OF QUICK START
MOTION is used to enable motion of any kind.
If MOTION is off, then no motion will be allowed.
MOTION is often connected to a stop button. Ground pin is to enable motion.

If MOTION is off during power-up, the VECTOR**STAR** will autobaud.

If your application does not allow you to use MOTION, then you must hardwire it on. If you are not using Connector C7, you can hardwire MOTION, by installing a jumper directly on C7 between Pins 15 and 16. The VECTOR**STAR** is shipped from the factory with this jumper installed.

- GATE Pin 17

The GATE input starts precalculated motion profiles. It is used to synchronize motion with external events. This line may be used as a general purpose input if its primary function is not used.

- O7, O8 Pins 19, 21

O7 and O8 are general purpose outputs.

- CYCLE READY Pin 23

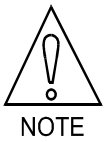
CYCLE READY is an output that indicates that the VECTOR**STAR** is ready for the CYCLE line to be activated. CYCLE READY is often connected to PLC's or to a lamp on an operator panel.

2.6.4.11 Wiring C8, Optional I/O

C8 is a 50-pin ribbon cable connector as shown in Figure B.7. It provides non-isolated, 5 volt TTL-level inputs and outputs are open collector and directly interfaced to 24-line, OPTO-22 compatible I/O boards. These boards are widely available and use industry standard optical isolation modules. This connector is optional. It is on top of the VECTOR**STAR**.

Note that you must provide a separate power supply when using standard OPTO-22 compatible I/O boards. This supply must provide 5 volts DC to power the I/O modules on the VECTOR**STAR** side of the isolation. The common of this supply will normally be connected to the common of the VECTOR**STAR** through the even numbered pins on Connector C7. Do not use the VECTOR**STAR** 5 VDC power supply to power I/O modules!

Additionally, a second power supply may be needed to provide power for those devices which are isolated from the VECTOR**STAR** by the I/O modules. The common of this supply should not be connected to the VECTOR**STAR** Common.



You must provide an additional power supply for the I/O modules.

- O1-O6 Odd numbered Pins from 37 to 47
  O1-O6 are general purpose outputs.

- STATUS Pin 35

The STATUS output indicates the status of the VECTOR**STAR**. You can configure STATUS to indicate "active" or "ready to be activated" with the software switch STATMODE. See the Enable and Fault Logic Diagram in Chapter 8 (Figure 8.10).

The state of the STATUS output is undefined for up to 25 milliseconds on power-up.

STATUS may turn on for up to 25 milliseconds after power-up.

- MANUAL Pin 33

MANUAL is used to change the VECTOR**STAR** from AUTO mode to MANUAL mode. For more information, refer to User's Manual Chapter 10. MANUAL may be used as a general purpose input if its primary function is not required.

- I1-I16 Odd numbered Pins from 1 to 31

I1-I16 are general purpose inputs.

- Power Meter Pin 9

2.6.4.12 J1 Configuration Jumper

Jumper J1 connects the RS-485 line terminators. Refer to Section 2.6.4.8, "Line Termination," for information.

2.6.5 Establishing Communications

Kollmorgen supplies a communications package called, Motion Link, that is designed especially for communicating with the VECTOR**STAR**. Other terminal emulators can also be used as long as the required data format is followed. This manual is written for use with Motion Link. For more information on Motion Link, refer to Chapter 7.

2.6.5.1 Required Data Format

The VECTOR**STAR** and your terminal must use the same format for serial data. RS-232-C and RS-485 describe hardware only. There are other specifications left to up to the user: full- or half-duplex, the number of bits per character, whether or not parity is used, the number of stop bits, and the baud rate. Full-duplex means both the terminal and the VECTOR**STAR** can send and receive at the same time. Half-duplex means only one system can talk at a time. Bits per character refers to the number of actual data bits sent at one time. The parity is a bit sent for error detection. The number of start and stop bits sets a minimum delay between characters. The baud rate is the rate at which bits are transmitted and received.

The VECTOR**STAR** requirements are:

Full-duplex
8 bits per character
No parity
1 start bit
1 Stop bit
Baud rates equal 300, 600, 1200, 2400, 4800, 9600, or 19200.

Most terminals and computers will allow you to set these options with little difficulty. Motion Link sets these requirements automatically.

2.6.5.2 First Transmission

Before you attempt to establish communications, you must:

1. Mount the VECTOR**STAR** and PA system as described earlier in this chapter.

2. Make all connections as described in the earlier sections of this chapter, especially connecting the VECTOR**STAR** serial connector to your terminal or computer.

2.6.5.3 Checking the Control Power

SHOCK HAZARD!

230 VAC is present on the Control Power. Be very careful when measuring these voltages.

Control Power is the 230 VAC connection at PA Connector "Control Input," pins 1 and 3.

Table 2.6 PA "Logic Output" Voltages

+15 Volts ±20%	Pins 1 and 5
-15 Volts ±20%	Pins 2 and 6
Common	Pins 3 and 7
+8 Volts ±20%	Pins 4 and 8

Turn off Control Power. Remove PA Connector "Control Input." Turn on Control Power. Use an AC voltmeter to confirm that the voltage on PA Connector "Control Input," Pins 1 and 3, is 230 VAC +/- 15%.

Turn off Control Power. Re-install PA Connector "Control Input." Remove PA Connector "Logic Output." Turn on Control Power. Use a DC voltmeter to confirm the logic supply (measure the pins in the PA Connector "Logic Output").

If none of these voltages are present, check control power (230 VAC) on PA Connector "Control Ouput," Pins 1 and 3. If one or more of the voltages are missing, one or more fuse in the PA may be open. Refer to the spare parts lists in Chapter 5 for descriptions of the fuses.

Turn off Control Power. Re-install PA Connector "Control Input."

Turn on Control Power. The green CPU LED on the front of the VECTOR**STAR** should begin flashing, about 10 times per second. This indicates that the VECTOR**STAR** is autobauding to establish communications.

If the CPU LED is not blinking...

Check the logic voltages from Table 2.6 above. If one or more of the voltages are missing, a fuse in the PA opened because the logic supply was miswired. You may have miswired either Connector "Logic Output" on the PA or Connector C4 on any VECTOR**STAR**. Check your wiring carefully.

Make sure the MOTION input is off (contacts are open) when you power-up the VECTOR**STAR**. If the contacts are closed, open them and power-down the VECTOR**STAR** and immediately power-up again. If the CPU LED is still not blinking, contact the factory.

1. If you are using Motion Link with an IBM-PC, type "ML" from DOS to start Motion Link. Motion Link will establish communications. Refer to Chapter 7 for more information on installing and running Motion Link.

2. Whether you are using a terminal or a PC, press the return key 5 to 10 times, pressing the key about twice per second.

These steps cause carriage return characters to be sent to the VECTOR**STAR**. When autobauding, the VECTOR**STAR** looks for these characters and uses them to determine the baud rate or speed at which your terminal is transmitting. Once the baud rate is determined, the VECTOR**STAR** can establish communications with your terminal or computer.

Then the VECTOR**STAR** should print this message (or one similar to it) on your terminal:

```
MOTION LINK BDS5/VS1  V4.0.0
(C) 1996 KOLLMORGEN MOTION
TECHNOLOGIES GROUP
—>
```

The green SYS OK LED on the front of the VECTOR-**STAR** should turn on and remain on at all times after power-up (and autobauding). The "—>" means that your VECTOR**STAR** is in the "interactive mode" and is ready to accept commands.

2.6.5.4 If You Can't Communicate...

If the VECTOR**STAR** does not respond, then check to be sure that Connector C5 is wired properly. A common mistake is that of having the transmit and receive lines swapped. Also, make sure that none of the wires in the cable are broken.

If the CPU LED has stopped blinking...

the VECTOR**STAR** has established communications. However, the transmission from the VECTOR**STAR** is not being displayed on your terminal. Check carefully for miswired cable, broken wires, or possibly a shorted transmit line from the VECTOR**STAR** to your terminal.

If the CPU LED is still blinking...

the VECTOR**STAR** has not established communications. You can determine that your terminal is working by disconnecting the serial cable at your terminal. Then, temporarily connect the transmit and receive pins to each other (usually Pins 2 and 3 on a 25-Pin DB-25 connector) and press a few keys. The characters that you type should appear on the terminal screen. If they do not, then your terminal is not functioning properly.

If your terminal is functioning properly...

the serial communications cable may have open circuits (broken or missing wires) or may not be connected to the proper pins. You can determine if your serial cable has open circuits by connecting your terminal on one end and disconnecting the cable on the VECTOR**STAR** end. Again, temporarily connect the transmit and receive pins (this time on the far end of the cable), then type on the keyboard. The characters that you type should appear on the terminal screen. If they do not, then your cable is not functioning properly.

If your cable is functioning properly...

the serial cable may still be the problem. You can determine if your cable is wired properly with an oscilloscope. For RS-232 systems, disconnect the cable from Connector C5. Connect ground on the scope probe to Pin 5 of the DE-9 connector on the cable. Use the oscilloscope to monitor Pin 2, with the time base between 1 and 20 milliseconds/division and the voltage sensitivity to 5 volts/division. While monitoring Pin 2, press several keys on the keyboard. The oscilloscope display should show Pin 2 changing between ±9 volts every time you press a key. If not, check Pin 3. If Pin 3 is responding, then Pins 2 and 3 are reversed. Remove power and exchange Pins 2 and 3. If there is no response, then the terminal may not be properly connected.

If you do not have an oscilloscope...

you may be able to use a digital voltmeter. Many digital voltmeters are quite sensitive and can detect character transmissions. You should see some activity (change in voltage) on the meter display every time you press a key.

If you think the cable is correct, but you still cannot communicate...

your terminal may not be configured for the data format listed in Section 2.6.5. Check the manual for your terminal or computer (virtually all IBM-PC compatible computers conform to this format). If you are using a terminal, be sure your terminal is set properly.

2.7 INITIAL CHECK-OUT

This section will discuss the procedure for checking most of the wiring on the system before enabling your VECTOR**STAR**. Communications must be established with your VECTOR**STAR** before continuing.

In this section, the VECTOR**STAR** will be used to check wiring to most of the VECTOR**STAR** inputs and outputs. You will need to enter some VECTOR**STAR** commands. This section will use a few commands that are described in more detail elsewhere in this manual. The first command is the print command. You can print the discrete inputs. For example, after the VECTOR**STAR** has printed the "—>" prompt, you can type:

P LIMIT

and the VECTOR**STAR** will print 1 or 0. 1 means that the input is on, indicating that the contacts are closed. 0 means that the input is off, indicating that the contacts are open.

In this manual, instructions that you enter will be shown in italics and surrounded by a double line box. The response from the VECTORSTAR will be in plain upper case letters.

2.7.1 Checking Discrete Inputs

You can check the state of all of the discrete inputs with the print command. This process will be demonstrated with the hardware travel limit switch, LIMIT. If you are not using the LIMIT switch, you can substitute another hardware switch, such as CYCLE or HOME, available on Connector C2. In this case, substitute the words "CYCLE" or "HOME" for "LIMIT" in the following discussion.

1. Open the switch contacts.

2. Verify that the contacts are open. If you are using a Kollmorgen Input Module or an industry standard OPTO-22 compatible input module, there will be an LED on each input to indicate its state. The LED will be off if the contacts are open.

3. Use your terminal to enter:

P LIMIT

4. The VECTOR**STAR** should respond with "0" indicating that the contacts are open.

5. Close the switch contacts.

6. Verify that the contacts are closed. (LED should be on.)

7. Use your terminal to enter:

P LIMIT

8. The VECTOR**STAR** should respond with "1" indicating that the contacts are closed.

Repeat this process for each discrete input that you are using:

__I1	__I2	__I3	__I4
__I5	__I6	__I7	__I8
__I9	__I10	__I11	__I12
__I13	__I14	__I15	__I16
__CYCLE	__GATE	__HOME	__LIMIT
__MANUAL	__MOTION	__REMOTE**	

**Note that if a fault condition exists, the state (on or off) of REMOTE may not be available. In this case, the value of REMOTE will print as -1.

2.7.2 Checking General Purpose Outputs

You can check all of the general purpose outputs by turning them on and then off. Be careful. The procedure in this section will activate all of the general purpose outputs. Be sure that activating an output will not cause a hazard to personnel or damage to equipment.

Commands in this section will turn on all general purpose outputs. Be certain that this is not a safety hazard. Make sure this will not damage equipment.

1. Turn on O1 by typing:

O1 ON

2. Verify that the output is on. If you are using a Kollmorgen Input Module, or an industry standard OPTO-22 compatible input module, there will be an LED on each output to indicate its state. The LED will be on if the output is on.

3. Turn off O1 by typing:

O1 OFF

4. Verify that the output is off.

Repeat this process for each discrete output that you are using:

__O1	__O2	__O3	__O4
__O5	__O6	__O7	__O8

2.7.3 Cycle Ready

CYCLE READY cannot be checked without a program. This topic is discussed in Chapter 10, "User Programs." Later, when you are familiar with programming, you can write a program that turns CYCLE READY on. Check this output at that time.

2.7.4 Checking STATUS

Check STATUS later in this chapter when the VECTOR**STAR** is active. The STATUS output will turn on when you enable the VECTOR**STAR**.

2.7.5 Checking Encoder Output

Check the encoder output with a two-channel oscilloscope. Place Channel 1 on OUTA and Channel 2 on OUTA'. Rotate the motor by hand. You should see the

encoder output on your scope. Repeat this process for OUTB and OUTB' and for OUTZ and OUTZ'. Note that OUTZ changes state only at one point during a full motor revolution.

2.7.6 Checking Encoder Input

If you have encoder input, connect your encoder to Connector C1. The encoder should not be moving. Type:

P PEXT

Now rotate the encoder. Type:

P PEXT

PEXT should have changed when you rotated the encoder shaft. If the results are not what you expected, use an oscilloscope to verify that all four signals are reaching Connector C1.

Note that encoder equivalent inputs are not available if your system has analog input.

2.7.7 Checking Pulse Input (Optional)

If you have a pulse input option, connect your pulse input to Connector C1. Stop the pulse train. Type:

P PEXT

Now inject a few pulses. Type:

P PEXT

PEXT should have changed by the number of pulses you injected. If the results are not what you expected, make sure that you have set the pulse format switch properly. Then, use an oscilloscope to verify that all four signals are reaching Connector C1.

2.7.8 Checking Analog Input

Connect the analog input voltage to Connector C10. Adjust the voltage to zero volts. Type:

P VEXT

Wait a few seconds and type:

P VEXT

VEXT should not have changed or should have changed very little. Now raise the input voltage to a few volts. Type:

P VEXT

and VEXT should have changed much more than it did with the input zeroed.

2.7.9 Checking the Resolver

The VECTOR**STAR** provides a feedback loss circuit which is designed to detect broken wires; in general, it will not detect wiring errors. You should use the following procedure to verify feedback circuit.


WARNING

The feedback loss circuit is designed to detect broken wires. In general, it does not detect wiring errors.

Turn the motor shaft by hand until the R/D converter output is within the ranges listed below. You can display the R/D output by repeatedly typing:

P PRD

The standard R/D resolution is 12 bits. The resolution of your VECTOR**STAR** is listed as part of the model number, which is shown on the front of the drive and described at the beginning of this chapter.

Rotate the motor shaft by hand until PRD is in the target range listed below:

Table 2.7. Target of PRD vs. R/D Resolution

R/D Resolution	12-Bit	14-Bit	16-Bit
Target Minimum	0	0	0
Target Maximum	250	1000	4000

Now rotate the motor shaft approximately 1/4 revolution CLOCKWISE. PRD should now fall in the range listed below:

Table 2.8. Target of PRD Versus R/D Resolution After Clockwise 1/4 Revolution

R/D Resolution	12-Bit	14-Bit	16-Bit
Target Minimum	800	3500	12500
Target Maximum	1300	5500	22500

If PRD does not fall into the range as listed above, then either the resolver is improperly wired or the feedback circuitry is not functioning correctly. The first step is to check the cable.

2.7.10 Checking the Resolver Cable

Follow this procedure if you are experiencing problems with the resolver. Disconnect the resolver cable at both ends and use an ohm meter to verify the following resolver cable wiring table (Table 2.9).

The following procedure uses an oscilloscope to measure the amplitude of signals with a frequency of 8 kHz. Some digital voltmeters can measure the amplitude of signals at high frequencies (like 8 kHz). However, many meters are designed to read 60 Hz and cannot be relied upon to read 8 kHz signals.

If the cable is correct, check the resolver reference signal. Install the resolver cable at both ends. Connect an oscilloscope probe to Connector C3, Pin 4 (extend the probe with a short length of wire to reach inside the connector housing if necessary) and connect the probe ground to common (Connector C3, Pin 10). Apply control power (230 VAC). The reference signal should be between 11.8 and 12.0 V peak-to-peak (4.17 and 4.24 V RMS) at a frequency of about 8 KHZ. If necessary, adjust the reference amplitude with potentiometer R237. This potentiometer is located between Connectors C2 and C3 and can be adjusted from the front of the unit without disassembly.

If the reference signal is correct and the feedback circuitry is not working, your VECTOR**STAR** may be malfunctioning. Contact the factory.

Table 2.9. Goldline Resolver Cable Wiring

VECTOR**STAR** C3	Signal	Resolver Connector
1	Sine Low	B
2	Shield	No Connect
3	Cosine Low	C
4	Reference High	F
5	Shield	No Connect
6	Not Used	No Connect
7	Sine High	A
8	Shield	No Connect
9	Cosine High	D
10	Reference Low	E
11	Shield	No Connect
12	Not Used	No Connect
Flying Lead	Thermostat (Black)	T
Flying Lead	Thermostat (Black)	U
Flying Lead	Optional Brake (Blue)	N
Flying Lead	Optional Brake (Blue)	P
Flying Lead	Optional Tach (Black)	R
Flying Lead	Optional Tach (White)	S

2.7.11 Checking the AC Line Voltages

SHOCK HAZARD!

Large voltages are present on the AC Line. Be very careful when measuring these voltages.

Open the circuit breaker or remove the fuses in the AC Line.

Turn on the AC Line. Use an AC voltmeter to check and record the 1- or 3-phase line-to-line voltage at the circuit breaker or fuse holders. Turn off the AC Line. Note the model number of the VECTOR**STAR** unit and refer to the Model Number Figures in Chapter 1 to confirm correct AC Line voltage.

2.7.12 Checking the DC Bus Voltage

Remove the AC Line. Wait 5 MINUTES for the DC BUS to discharge.

WAIT 5 MINUTES FOR THE DC BUS TO DISCHARGE!

The DC Bus is connected to large capacitors inside the PA. These capacitors can store a lot of energy.

Remove the BUS+ and BUS- leads from the PA Power Terminal Block. Disconnect Connector C2 from the PA.

Turn on Control Power to PA Connector C1 and turn on the AC Line to the PA Power Terminal Block (L_a, L_b, and L_c).

SHOCK HAZARD!

Large voltages are present on L_a, L_b, L_c, BUS+ and BUS-. Be very careful when measuring these voltages.

Check and record the DC Bus output voltage at BUS+ with respect to BUS- on the PA Power Terminal Block. It should be approximately 325 VDC for 230 VAC line voltage, or 162 VDC for 115 VAC line voltage.

Remove the AC line voltage. Wait 5 minutes for the DC Bus to discharge.

Reconnect the BUS+ and BUS- leads to the Power Terminal Block on the PA. Be careful to reconnect the leads with the correct polarity. Re-install Connector C2 on the PA.

OBSERVE POLARITY OF THE DC BUS!

When interconnecting Industrial Drives GOLDLINE Series Products, connect BUS+ to BUS+ and connect BUS- to BUS-.

2.7.13 Checking the Motor

The MOTOR command is provided to ensure that your VECTOR**STAR** is properly configured for your motor. Type:

MOTOR

The VECTOR**STAR** should respond with something like:

MOTOR = B-204B

You can then verify that your motor is a 204B. Always check the motor nameplate to verify that you have wired the correct motor to your VECTORSTAR. This should agree with the part of your VECTORSTAR model number as described in Chapter 1.

Wiring the VECTORSTAR to a motor for which it is not configured can cause the system to become unstable. Verify that you are connecting the correct motor.

Verify that motor wiring is correct. It is very important that you wire the motor properly. Brushless motors are not like induction motors. You cannot simply interchange two phases to reverse the direction of rotation. You MUST connect Pin A of the motor connector to Ma of the VECTOR**STAR** power connector, Pin B of the motor connector to Mb, and Pin C of the motor connector to Mc.

Follow this procedure to check motor wiring:

1. Turn off the main AC Line. Leave Control Power "on."

2. Remove all loads from the motor. The motor must be able to rotate freely for this test.

3. Rotate the motor by hand until PRD is in the zero position as shown in Table 2.10. Note that the position must be less than the small number OR greater than the large number.

To display PRD, type:

```
P PRD
```

Table 2.10. PRD Range for "Zero Position"

R/D Resolution	PRD Less Than	PRD Greater Than
12-Bit	25	4070
14-Bit	100	16300
16-Bit	400	65000

4. Turn on (close contacts of) LIMIT, MOTION, and REMOTE.

5. Turn on Control Power only.

6. Put the VECTOR**STAR** into the zeroing mode. Note that zero mode is not used in induction (V-Series) motors. Type:

```
ZERO ON
```

Zero mode is not used in induction (V-Series) motors.

7. Turn on AC Line.

8. Enable the VECTOR**STAR**. Type:

```
EN
```

The motor may move a small amount, but PRD should remain in the zero position. Type:

```
P PRD
```

If the motor rotates to the zero position, then type:

```
DIS
ZERO OFF
```

and continue to the next section.

If the motor rotated to the wrong position, then you must stop and correct this problem. Either the resolver is improperly wired, the motor is improperly wired, the motor is not functioning properly, or the resolver is not functioning properly. In most cases, the wiring is the problem. Check your motor and resolver wiring carefully. Follow the procedure in “Checking the Resolver Cable” in section 2.7.10. If you have wired the motor through motor-overload relays, verify that the relay is closed. Contact the factory if your motor does not rotate to the zero position and you can not correct the problem.

DO NOT PROCEED IF YOUR MOTOR IS NOT ZEROING PROPERLY.

Figure 2.5 VectorStar Pinout Information

I/O Relay Outputs*	
Connector J12**	
Pin	**Assignment**
50	Out 1
48	Relay
46	Out 2
44	Relay
42	Out 3
40	Relay
24	Out 4
18	Relay
22	Out 5
20	Relay
14	Out 6
12	Relay
10	Out 7
8	Relay
6	Out 8
2	Relay
4	Motor OT

**Not available on VSA

*Max potential 28V @ 1 Amp

I/O Interface	
Connector C7 (J1)	
Pins	**Assignment**
1	Spare
3	Tach
5	Shld
7	I Monitor
9	+15
11	Remote Rtn
13	Cycle Rtn
15	Cycle
17	Remote
19	O1 Hi
21	Spare
23	Com
25	Shld
Even Pins	Com

I/O Interface*	
Connector C8 (J20)	
Pins	**Assignment**
1	IN16
3	IN15
5	IN14
7	IN13
9	IN12
11	IN11
13	IN10
15	IN9
17	IN8
19	IN7
21	IN6
23	IN5
25	IN4
27	IN3
29	IN2
31	IN1
33	Manual
35	Status
37	O6
39	O5
41	O4
43	O3
45	O2
47	O1
49	Meter Out
50	Common
2	Common

*Even pins common

Encoder Equiv. Out	
Connector C1 (J7)	
Pins	**Assignment**
1	A
6	A
2	B
7	B
3	Z
8	Z

Analog Input	
Connector C10 (J250)	
Pins	**Assignment**
5	Diff Hi
6	Diff Lo
7	Com
8	Gnd

Customer Interface	
Connector C2 (J9)	
Pin	**Assignment**
1	Spare
2	Tach
3	Shld
4	I Monitor
5	+15
6	Remote Rtn
7	Cycle Rtn
8	Cycle
9	Remote
10	O1 Hi
11	Spare
12	Com
13	Shld
14	Com
15	Com
16	Relay Contact
17	Relay Contact
18	Home Rtn
19	Home
20	O1 Lo

Resolver Feedback	
Connector C3 (J13)	
Pin	**Assignment**
4	Ref Hi
10	Ref Lo
7	Sine Hi
1	Sine Lo
3	Cos Lo
9	Cos Hi
5	Shld
8	Shld
2	Shld
11	Shld
6	Spare
12	Spare

Note:

C = Low power unit

J = High power unit

CHAPTER 3
QUICK START

3.1 INTRODUCTION

Here, we assume that you are experienced with servo drives and their installation. The safety precautions are not included in this chapter. However, all safety measures should be properly followed before permanently installing this equipment.

3.2 QUICK INSTALLATION FOR LOW POWER UNIT

For the following instructions, refer to Figure 3.1 on page 35 and Figures B.1 and B.2 in Appendix B. For more information on each step, please refer to the section listed in parentheses.

1. Mount the drive and power supply (2.5.1 and 2.5.2).
2. Wire the external regen resistor to the power supply (2.5.3).
3. Connect DC bus between the power supply and drive (2.6.2.4).
4. Connect motor power (Ma, Mb, and Mc phase must be correct) (2.6.2.1) and resolver cables (2.6.4.6).
5. Wire the ground (drive, power supply and motor) (2.6.1).
6. Wire the 3-phase AC line input (2.6.2.5).
7. Connect 230-volt to “Control Input” of PA28/50/70/85 (pins 1 and 3) (2.6.3.1).
8. Connector C6 of the drive is for the cooling fan: 230 volts for an AC fan and 24-28 volt DC source for a DC fan (2.6.4.9).
9. Connect “Logic” of PA28/50/70/85 to C4 of the drive control (logic) power (4 wires) (2.6.3.3).
10. Connect an RS232 cable to a PC serial port, see Figure B.4 (2.6.4.8).
11. Place two jumpers between pins 11 and 12, 15 and 16 of C7, which is on the top of the drive, to close MOTION and LIMIT. (For temporary setup only.)
12. Close REMOTE by connecting pin 9 to 5 and pin 6 to 15 of C2 on the drive. (For temporary setup only.)
13. Apply control (logic) and main power to the system. The BUS, CPU and SYSOK lights should be on.
14. Run “Motion Link” from the PC (if you have a Windows version, use Terminal mode), press ENTER or Esc to establish communication.

15. Turn motor by hand, read rotor position by issuing "P PRD" repeatedly, which shows the resolver feedback. If you have a 12-bit resolution setting for R/D, you should see the count change from 0 to 4095 when the rotor turns one revolution.

16. Type "EN"; the drive should be energized and you should hear a slight 10kHz sound from the motor.

17. Type "J 10"; the motor will run at 10 RPM. You can check it by typing "P VFB" to read the speed.

18. Type "J 1000" to make the motor run at 1000 RPM.

19. Type "J 0" and the motor will stop.

20. Type "K" to disable the drive after motor stand-still..

If you receive an error code, check Appendix C.

Note that we have omitted the power supply relay connection and motor thermostat connection. We also assume that you do not need encoder input, encoder output, or general I/O for the moment. For quick start purposes, the REMOTE input is temporarily using a non-isolated power supply directly from the drive (pins 5 and 15 of C2). MOTION and LIMIT also must be connected where needed.

3.3 ANALOG INPUT CONTROL

If you want to try the analog input control, follow these steps:

1. Connect analog voltage input to pins 5 and 6 of C10 on the drive (top 2 pins on the left column)..

2. Type "VSCALE=10000" to scale analog input to 10000 RPM/10 volts. As an alternative, from Motion Link, check the values of GEARI and GEARO by typing "P GEARI" and "P GEARO." GEARO should be 16384 and GEARI should be 700 for a system of 10 volts input = 10000 RPM, or 420 for a system with 10 volts input = 6000 RPM.

3. Type "A2D=1" to turn on analog input. Type "GEAR=1" to turn on Gear function.

4. Give zero volt input and enable the drive by typing "EN."

5. If there is a drift, you may increase the value of Z0, which is the deadband, 1 equals 620 micro volts.

6. If the analog input is 1 volt, the motor should run at 1000 RPM for a 10000 RPM system or 600 RPM for a 6000 RPM system.

At this point, you have completed this quick start section by performing two of the simplest operations: JOG and analog input (GEAR must be on). Advanced functions for the VECTOR**STAR** are described in detail throughout this manual.

3.4 QUICK TEST FOR SPINDLE FUNCTIONS

If you want to test the spindle functions built into the drive, and the drive has been connected to a CNC, these instructions may help you quickly test the system.

1. Make sure the connections between the drive and the CNC are correct (see the table in Figure 3.1, note I4 and I5 are negative logic).

2. Make sure LOAD=1.

3. Using Motion Link, check the input status by pressing the Esc key first to get the prompt "=>" and typing "P IN"; you should get a reading of 24. You can check each input by typing "P I1," "P I2," "P I3," ..., "P I7."

4. If everything is okay, the motor should be enabled by giving I7=1 from the CNC.

5. Check the motor speed by giving an analog signal to the drive and typing "P VEXT" to check if the drive receives the input.

6. You may repeat the test of adjusting the deadband of analog input by changing Z0.

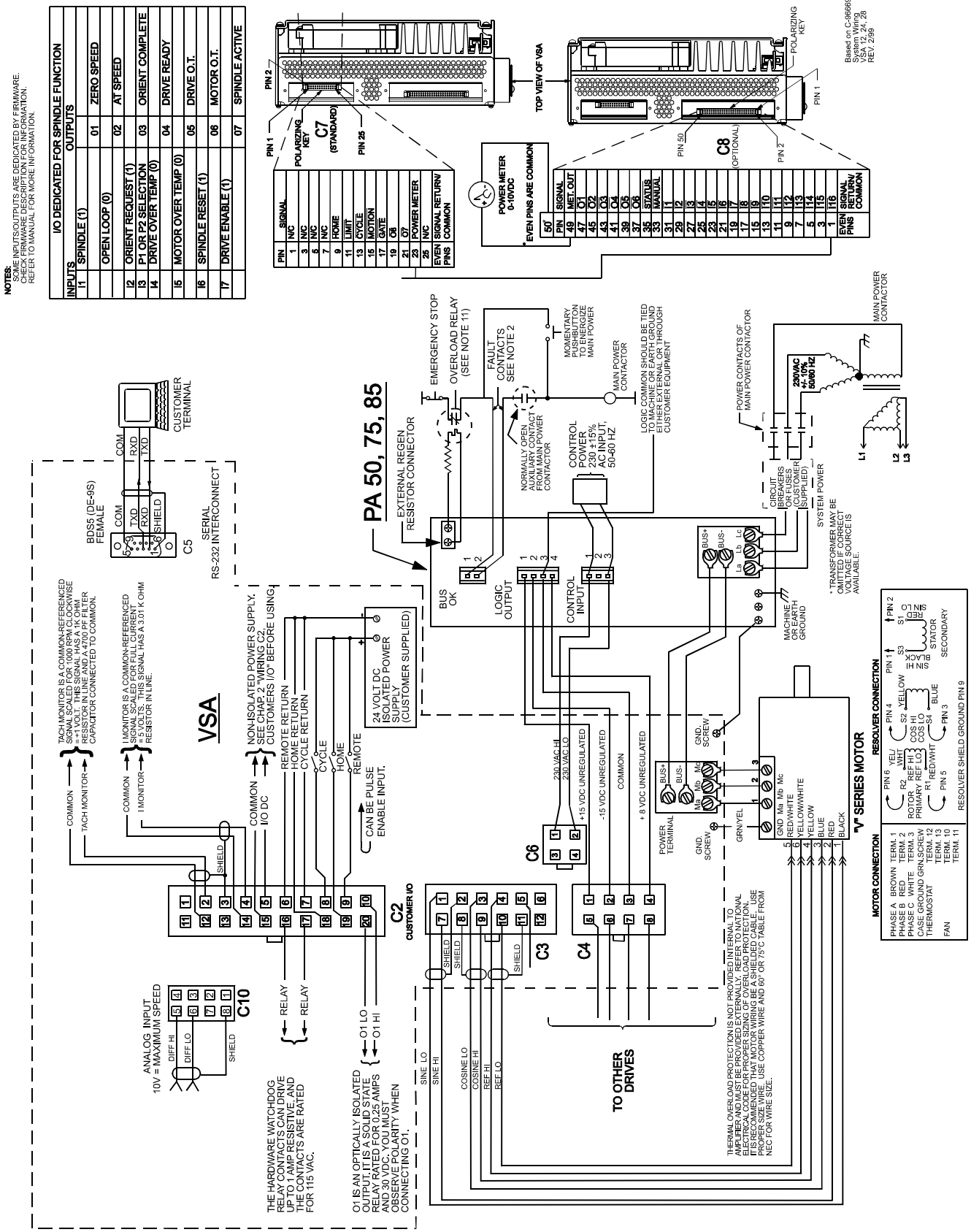


Figure 3.1 System Wiring VSA

CHAPTER 4
OPERATION

4.1 INTRODUCTION

The information in this chapter will introduce you to the system components and their dependence upon one another. Also, it will help you ensure each component is configured and functions properly. At this point, all safety stops and other precautions should be in place and working properly. Be prepared to stop the machine if necessary.

In addition, this chapter will enable you to compensate your motor for load conditions. Tuning is an important step in setting up and maintaining your VECTOR**STAR** servo system. This chapter defines and explains tuning in detail. A flowchart is also provided for easy step-by-step instructions to tune the servo system.

4.2 START-UP AND CHECKOUT

You should now be ready to supply power to test the servo systems functions and features. Work with only one axis section at a time. Confirm all other VECTOR**STAR** amplifiers are inhibited, meaning the enable circuits are open (high).

Appropriate precautions should be taken to stop the machine if necessary. Limit switches and safety devices should be in place.


WARNING

THE MOTOR MAY MOVE UNEXPECTEDLY!

BE PREPARED TO DISABLE THE VECTORSTAR!

Commands in this section will enable the VECTORSTAR. The system may be unstable. The motor may begin oscillating or run away. Be prepared to disable the VECTORSTAR quickly by turning off (opening the contacts of) LIMIT or REMOTE.

This section discusses how to enable the VECTOR**STAR**. Follow this procedure:

1. Turn on (close contacts of) LIMIT, MOTION, and REMOTE.
2. Turn on Control Power.
3. Turn on the AC Line.
4. Enable the VECTOR**STAR**. Type:

```
EN
```

The motor should be still. If your motor is oscillating, disable the VECTOR**STAR** by typing:

```
DIS
```

4.2.1 ERROR 17, FEEDBACK LOSS

If the VECTOR**STAR** generates ERROR 17, FEEDBACK LOSS, then a lead in the resolver cable is probably broken. First, verify that the cable is wired correctly using the procedure in "Checking the Resolver Cable" in section 2.7.10.

If the cable appears to be wired correctly, use an oscilloscope to verify that the sine (Connector C3, Pin 7) and the cosine (Connector C3, Pin 9) are present. Note that the amplitudes of both these signals vary with motor position, so you will need to rotate the motor by hand to force sine and cosine to their maximum values. Expect a sine wave with a maximum peak-to-peak level of about 6.5 Volts at 8 kHz.

If both sine and cosine are present on the connector and your VECTOR**STAR** continues to generate ERROR 17, your unit may be malfunctioning; contact the factory.

4.2.2 ERROR 14, POWER BUS

ERROR 14, POWER BUS, is generated when the VECTOR**STAR** detects either an undervoltage or an overvoltage. If the VECTOR**STAR** cannot be enabled because ERROR 14 continually reoccurs, it is probably because of undervoltage. Return to "Checking the AC Line Voltage" and "Checking the DC BUS Voltage," beginning in section 2.7.11, to ensure that the DC Bus is present.

If this error occurs only when the system is powered up, it is probably because your program attempts to enable the VECTOR**STAR** before bus voltage is present. OK2EN is a switch that is ON when your VECTOR**STAR** can be enabled without generating a fault. You can delay enabling the VECTOR**STAR** until bus voltage is present by modifying your program to wait for OK2EN to be ON.

```
TIL OK2EN EQ ON
EN
```

If the error occurs occasionally, it could be overvoltage or undervoltage. Overvoltage is usually caused by regenerative energy from a deceleration which forces the DC BUS voltage to rise above the VECTOR**STAR** overvoltage level—about 200 VDC for 115 VAC line voltage systems and about 400 VDC for 230 VAC. If the error occurs only during deceleration, it is probably an overvoltage error. This can be corrected by reducing the deceleration rate (although often, it must be reduced dramatically) or by adding increased regeneration capability. Contact Kollmorgen Application Engineering to add regeneration capability.

Undervoltage is caused by the loss of the AC Line. The undervoltage detection level is set well under 100 VDC (70 VAC) so that low line (from "brown out") almost never causes an undervoltage fault. Your system may include protective circuits that remove power from the VECTOR**STAR** when a problem is detected elsewhere in the system. This can cause the VECTOR**STAR** to generate ERROR 14 and lead you to suspect the VECTOR**STAR** of causing the original problem. If the problem occurs only rarely, you may have to purchase or rent a device to monitor the DC BUS voltage to determine the cause.

4.2.3 If Your VECTORSTAR System is Unstable...

If the motor was oscillating, you need to retune your system. First, try to stabilize the system with the TUNE command. Type:

```
TUNE 10 2
```

Enable the drive. If your system is stable, you can skip ahead to the next section. If you want to improve the response, see Section 4.5 for details on the TUNE command.

If your system is not stable, disable the VECTOR**STAR**. You need to detune the system. First, disable the position loop. Type:

```
PL OFF
```

Disable the integrating velocity loop (in other words, enable the proportional velocity loop). Type:

```
PROP ON
```

On power-up, the integrating velocity loop will be enabled (that is, PROP is turned off) and the position loop will be turned on. Be careful to turn PROP on and PL off every time you power-up until you stabilize the system for an integrating loop.

PROP is turned off and PL is turned on at power-up. If you are following this procedure to stop oscillations, be certain to turn PROP on and PL off every time you power-up the drive.

Now enable the drive by typing:

```
EN
```

Reduce the proportional gain (KPROP) until oscillations stop. Type:

```
KPROP = KPROP/2
```

You may need to repeat this command a few times.

After oscillations stop, enable the position loop by typing:

```
PL ON
```

Next, reduce the position loop gain (KP) until oscillations stop. Type:

```
KP = KP/2
```

You may need to repeat this command a few times.

If you need to tune your system for better performance, see Section 4.3.

4.2.4 Jogging the Motor

If you can enable the VECTOR**STAR** without motor oscillations, then you can jog the motor. First, you should temporarily disable the VECTOR**STAR** software position limits. Type:

```
PLIM OFF
```

Now type:

```
J 10
```

The motor should rotate at 10 RPM. If PROP is on, the motion will be very unsteady. You need an integrating velocity loop for good low speed performance.

Disable the VECTOR**STAR** and enable the software position limits. Type:

```
DIS
PLIM ON
```

4.2.5 Low Speed Adjustment

SKIP THIS SECTION IF YOU HAD TO TURN PROP ON TO STABILIZE YOUR SYSTEM. The VECTOR**STAR** has a low-speed adjustment. This adjustment helps smooth very low-speed motion. This adjustment is made with the variable KC which is normally set to 200. This value of KC produces sufficiently smooth low speed performance for almost all applications. However, if your application is very demanding, you may want to adjust KC somewhat. It is rare for this procedure to be required.

1. Turn PL off by typing:

```
PL OFF
```

2. Twist the motor shaft back and forth lightly at about 1 or 2 twists per second. You should feel very slight “graininess.” This graininess is similar to the feel of anti-backlash gears. If you want to make the graininess more pronounced (so that you can feel it), set KC to zero. Type:

```
KC = 0
```

3. Adjust KC so that the graininess is minimized. The best way to do this is by attaching a mirror to the motor shaft and shining a laser onto the mirror and observing the reflected dot about 10-20 feet from the motor. You can attempt to “feel” the graininess, but that measurement is so coarse that you should probably just set KC to 200 and skip this procedure. The normal range for KC is between 175 and 225. This adjustment must be repeated when either the motor or the amplifier is changed out.

4. Restore PL. Type:

```
PL ON
```

This completes the initial check-out.

4.3 SYSTEM COMPENSATION

Feedback systems (like a motor controller) require tuning to attain high performance. Tuning is the process whereby the position and velocity loop gains are set, attempting to optimize the performance of a system (a VECTOR**STAR** and a motor connected to a load) to a three-part criterion:

- Noise Susceptibility
- Response
- Stability

Tuning is normally a laborious procedure requiring an experienced person. However, the VECTOR**STAR** provides many tools to aid tuning, making it a much simpler process.

In a broad sense, the performance of a system is characterized by its noise susceptibility, response, and stability. These quantities tend to be mutually exclusive. The system designer must decide what noise susceptibility (in the form of a "busy" motor) is acceptable.

"Busyness" is random activity in the motor and can often be felt on the motor shaft. Busyness in a motor should not be confused with PWM noise. PWM noise is high-pitched, relatively constant noise and cannot be felt on the motor shaft.

Response is a measure of the system's quickness. Response can also be characterized by bandwidth and by rise time in response to a step command. Normally, designers want high bandwidth, though sometimes the response is purposely degraded to reduce stress on mechanical components. This is called *detuning*. Typical velocity loop bandwidths range from 20 to 60 Hz. Typical position loop bandwidths range from 0.1 to 0.2 times the velocity loop bandwidth.

Stability measures how controlled the system is. Stability can be measured with damping ratio or with overshoot in response to a step command. A discussion of different levels of stability follows.

4.3.1 Critical Damping

Generally, the most desirable amount of damping is Critical Damping. Critically damped systems respond as fast as possible with little or no overshoot. In Figure 4.1, the graph shows the response of a TACH signal (on Connector C2, Pin 2) to a square wave input when the system is critically damped.

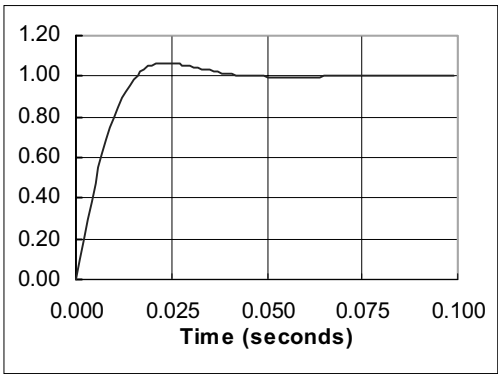


Figure 4.1. Critical Damping

4.3.2 Underdamping

Sometimes the system is tuned for critical damping and the system is still too slow. In these cases, you may be willing to accept less than critical damping. For applications that can work properly with a slightly underdamped system, you may reduce the stability to improve the response. The graph in Figure 4.2 shows a slightly underdamped system.

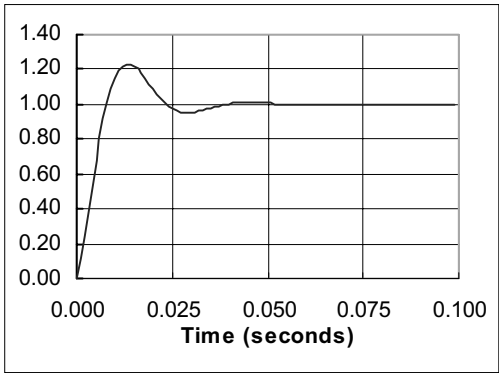


Figure 4.2. Underdamping

4.3.3 Overdamping

An overdamped system is very stable but has a longer response time than critically damped or underdamped systems. Also, overdamped systems are noisier than less damped systems with the same response rate. The graph in Figure 4.3 shows an overdamped system.

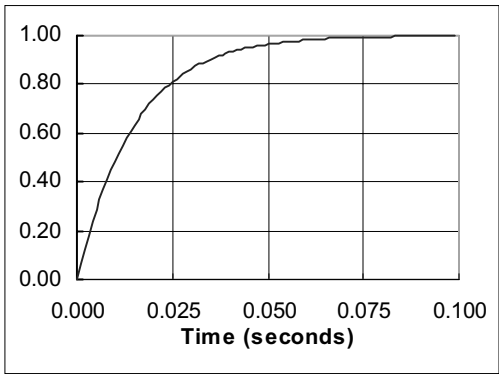


Figure 4.3. Overdamping

4.3.4 Ringing

When you are tuning the VECTOR**STAR** you may tune it so that the response rings. Ringing is caused when you attempt to tune the VECTOR**STAR** for either too rapid response (too high bandwidth) or too much stability (too much damping) or both. The only solution is to reduce the bandwidth or the stability or both. In Figure 4.4, the graph shows a system that rings.

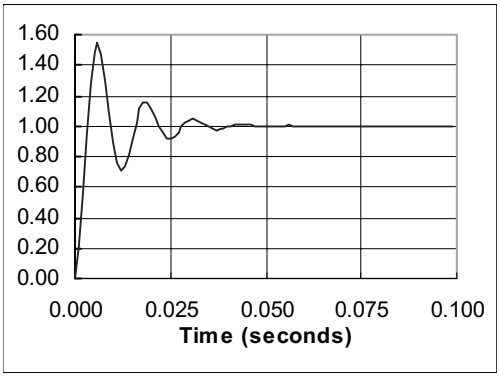


Figure 4.4. Ringing

4.4 TUNING

The TUNE command shakes the motor vigorously. Secure the motor before tuning.

The VECTOR**STAR** is usually shipped with a tuning that will work reasonably well with the load inertia between 0 to 4 times the rotor inertia. Many applications have approximately matching inertia. If your system does, you may not need to adjust the tuning of your VECTOR**STAR**. The following section describes how you can re-tune your system.

When tuning a system, it may be desirable to disable the VECTORSTAR quickly. You can use K, the one-letter KILL command, to disable your system.

The VECTOR**STAR** provides self-tuning. This is a feature that senses the inertial load of your system and then attempts to set tuning parameters accordingly. Note that self-tuning is not fool-proof. You may need to adjust one or two of the tuning parameters to get exactly the response you need.

THE MOTOR MAY OSCILLATE!

Unloaded motors tuned for a large inertia load may become unstable when the system is activated. If the system becomes unstable, remove the power immediately.

4.4.1 If Your System Is Completely Unstable...

If your system is completely unstable when you enable it, remove power immediately. After restoring power, but before enabling the VECTOR**STAR**, turn off the switch PL, reduce KV to 100, and reduce KVI to 0. This should make the system stable.

```
;TYPE THESE LINES ONLY IF YOUR
;VECTORSTAR IS UNSTABLE WHEN YOU
;ENABLE IT.  DON'T FORGET TO RESTORE
;PL WHEN YOU HAVE FINISHED TUNING.
PL  OFF
KVI = 0
KV = 100
```

If the VECTOR**STAR** is still unstable, remove power and contact the factory. If it is stable, continue on with tuning. Do not forget to turn PL back on when you have finished tuning. Also, PL is always turned on during the VECTOR**STAR** power-up.

4.4.2 Reducing ILIM

You may need to reduce ILIM before executing the TUNE command since the TUNE command causes the motor to "shake" at about 15 Hz and at full torque. This may damage some machines. Also, lightly loaded motors can overspeed if ILIM is too high. You should raise ILIM to the highest level that does not cause problems, because the tuning may not be acceptable if ILIM is too low. The effect can be that the torque the VECTOR**STAR** produces is "swamped out" by friction. If you are not sure how much ILIM is necessary, reduce ILIM to a low value (say 5 or 10%) and gradually raise it. If the tuning is acceptable (that is, it does not ring or overshoot excessively, and it does respond fast enough), then you are done. Do not forget to restore ILIM to its original value.

The TUNE command shakes the motor vigorously. You may need to reduce ILIM before executing the TUNE command to protect your machine. Do not forget to restore ILIM when tuning is complete.

The TUNE command can cause the motor to overspeed. You may need to reduce ILIM to prevent overspeed errors. Do not forget to restore ILIM when tuning is complete.

4.5 TUNE COMMAND

When you enter a TUNE command, you specify the response time and the stability level. The response time is specified in the form of bandwidth. The higher the bandwidth, the faster the response. The level of stability is specified as follows:

1. Slightly overdamped
2. Critically damped
3. Slightly underdamped

The drive will be enabled and the motor will turn. Make sure the motor is secured. Even if the VECTORSTAR is disabled, it will enable long enough to execute the TUNE command.

Enable the VECTOR**STAR** and type this command:

TUNE 30 2

The VECTOR**STAR** will shake the motor and set the tuning so that the velocity loop has a bandwidth of approximately 30 Hz and is critically damped. The allowed bandwidths are 5, 10, 15, 20, 25, 30, 40, and 50 Hz.

The TUNE command only works when PDF = 1. When PDF = 0, a PI controller is used. For the PI controller, KPROP represents the proportional gain and KVI represents the integrating gain. If PDF is on, autotuning will change the values of KV and KVI.

The tune command does not always provide an acceptable tuning. If not, you can tune the VECTOR**STAR** yourself.

4.6 TUNING THE VECTORSTAR YOURSELF

If you use the TUNE command, and the resulting tuning variables cause the system to oscillate, there are generally two reasons:

1. The bandwidth in the TUNE command is set too high for the system to function properly.

2. The low-pass filter is set too low (this only applies if LPF is on).

In either case, first raise the low-pass filter frequency (LPFHZ) to as high a level as is acceptable. You may even decide to remove it by setting LPF to off.

If the TUNE command does not provide a suitable set of tuning variables, then you have the option of tuning the VECTOR**STAR** yourself. You will need an oscilloscope. Connect an oscilloscope channel to TACH MONITOR on Connector C2, Pin 2; attach the scope ground to COMMON on Connector C2, Pin 14. Use the TUNE command to get as close as possible.

4.6.1 Tuning the Velocity Loop

The drive will be enabled and the motor will turn. Make sure the motor is secured.

The flow chart in Figure 4.5 shows how to manually tune an integrating velocity loop. This procedure sets KV and KVI. First, you should use the TUNE command to set KV and KVI close to optimum values. Apply DC bus voltage to the VECTOR**STAR**. Follow the instructions shown the flow chart. The motor should start and stop every second. Press the escape key to enter the Monitor mode where you can change tuning constants. The tach should be on the oscilloscope, showing the motor performance. As the drawing notes, you should increase KV for increased stability and increase KVI to make the system more responsive.

You need to make several decisions: Is the unit underdamped? Is the system response too fast? Is the system ringing? Is there a resonance present? Then, take the action listed in Figure 4.5

There is a close relationship between the response of the system and the variable KVI. Response is often measured by the system *bandwidth*. Bandwidth is the frequency with which the system response falls to 70% of the nominal response. For example, if your velocity command was a sine wave with peaks of ±100 RPM, the bandwidth would be the frequency that the response fell to a sine wave with peaks of ±70 RPM. The relationship between velocity loop bandwidth and KVI is shown in Table 4.1.

Table 4.1 Velocity Loop Bandwidth vs. KVI

KVI	VELOCITY LOOP BANDWIDTH
1400	5 Hz
2650	10 Hz
4000	15 Hz
5000	20 Hz
6250	25 Hz
7500	30 Hz
8750	40 Hz
10000	50 Hz

If you are using a proportional velocity loop (PROP is on), then adjust KPROP until the motor is performing appropriately.

4.6.2 Tuning the Position Loop

Once the velocity loop is tuned, you can tune the position loop. Break program execution and stop motion by typing S. Type in the following commands:

The drive will be enabled and the motor will turn. Make sure the motor is secured.

```
PEMAX  30000
ZPE              ;ZERO POSITION
                 ;ERROR TO AVOID
                 ;POSITION ERROR
                 ;OVERFLOW WHEN
                 ;ENABLING POSITION
                 ;LOOP
PL  ON
KF=0
RUN  80
```

The motor should again begin turning. Now adjust KP until the motor is performing appropriately. Table 4.2 shows the relationship between a properly tuned position loop (that is, the highest setting for KP) and velocity loop bandwidth. Note that the position loop bandwidth will be substantially lower than the velocity loop bandwidth (usually by a factor of 5 to 10).

Table 4.2 Velocity Loop Bandwidth vs. KP_{MAX}

KP_{MAX}	VELOCITY LOOP BANDWIDTH
500	5 Hz
1000	10 Hz
1500	15 Hz
2000	20 Hz
2500	25 Hz
3000	30 Hz
4000	40 Hz
5000	50 Hz

If you want to eliminate some or all of the following error, you can raise KF as high as unity feed-forward (Unity is defined as KF = 16384). However, the larger you make KF, the more you must reduce KP to eliminate overshoot and thus reduce the position loop performance. If you cannot get the desired performance from the position loop, then try reducing ACC and DEC to reduce overshoot. This can be a good way to limit overshoot in the position loop, and you may be able to raise KP slightly (about 20%) to improve performance.

4.7 ANALOG INPUT

If the drive needs to be configured as an analog input device, turn on the GEAR and A2D switches. For scaling the input command, use variable VSCALE. The units are RPM at 10 volts input. After VSCALE is assigned, VXNUM and VXDEN will be changed accordingly to this scale.

The analog input signal should be within -10 to +10 volts and goes to J10. The top left of J10 is Diff-high and second down of left is Diff-low.

For instance, if you want 9.75 volts input generating 10000 RPM velocity command, you need VSCALE = 100000/9.75 = 10256. The drive will calculate the GEARI, GEARO, VXNUM, and VXDEN.

If you want 10 volts input generating 2000 RPM velocity command, just make VSCALE = 2000.

The GEARI, GEARO, VXDEN, and VXNUM can be assigned individually any time after VSCALE is assigned, but VSCALE will assign all of them..

4.8 RECORD AND PLAY

The RECORD command allows you to record most VECTOR**STAR** variables in real time for later playback. You can simultaneously record up to four variables. You can record any variable except PE, REMOTE, TMR1, TMR2, TMR3, TMR4, VAVG, VXAVG, or any user switches. You can specify the time between points from one millisecond to one minute. You can record up to 1000 instances of 1 variable, 500 instances of 2 variables, 333 instances of 3, and 250 instances of 4 variables.

The format of the RECORD command is:

RECORD <Number> <Time> <1 to 4 Variables>

where number is the number of intervals over which the variables will be recorded, and time is the time in milliseconds of each interval.

Note: <Number> <= 1000 for 1 Variable
<Number> <= 500 for 2 Variables
<Number> <= 333 for 3 Variables
<Number> <= 250 for 4 Variables

The following example records the velocity response of the VECTOR**STAR** to a JOG command.

```
405$                ;BEGINNING LABEL
EN                  ;ENABLE
                    ;VECTORSTAR
RECORD 500 1 VFB    ;RECORD VFB FOR
J 1000              ;1/2 SECOND JOG
B                   ;1000 RPM
```

After data is recorded, you can use the PLAY command to print each point on the screen. However, Motion Link provides all the routines to retrieve, plot, print, and store recorded data on your computer and line printer.

The RECORD command is useful when tuning a system because you can display the VECTOR**STAR** response to commands without an oscilloscope. However, it is not limited to tuning. For example, you can record VCMD to plot a motion profile, or you can plot VEXT to watch the external encoder/analog input. You can also plot user variables to watch the performance of your program.

4.9 PROBLEMS

Sometimes there are problems tuning. Usually the TUNE command will provide you with a tuning that is either acceptable or close to acceptable. If not, you can tune the system yourself. Sometimes there are physical factors that prevent you from attaining the performance you need. These problems fall into four categories:

1. Overloading the Motor
2. Compliance
3. Resonance
4. Changing Load Inertia or Reflected Inertia

4.9.1 Overloading the Motor

Overloading the motor is the most common problem for positioning systems (that is, systems with PL on). If you overload the system, the position error can grow to very large values. When the command stops, the motor “reels in” the following error and can overshoot excessively. It looks like a tuning problem, but it is actually caused by the motor being undersized, ACC or DEC being set too high, or ILIM being set too low.

When a motor is overloaded, it has the following characteristics:

- The system overshoots, sometimes excessively, but does not ring or oscillate.
- Reducing ACC and DEC eliminates the problem.
- Turning off PL eliminates the problem.
- The motor current is near or at saturation during a large part of the move. Use the RECORD function to record ICMD. If ICMD is equal to ILIM for more than a few milliseconds, then your system is saturated.

Overloading the motor can be corrected by the following actions:

- Reducing ACC and DEC.
- Reducing the load on the motor.
- Increasing ILIM (if it is less than IMAX).
- Using a VECTOR**STAR** with a higher current rating.
- Using a motor with more peak stall torque.

4.9.2 Compliance

In compliant systems, the load is not tightly coupled to the motor shaft. If you move the load by hand, you can feel springiness. Compliant systems often are very stable when you tune with lower target bandwidths. However, they oscillate vigorously at low frequencies when you try to tune them for higher bandwidths.

When a system is compliant, it has the following characteristics:

- There is springiness between the motor and the load or at the motor mounting plate.
- The TUNE command calculates tuning variables that cause the system to oscillate.
- The frequency of oscillation is less than 100 Hz.

Compliance can be corrected by the following actions:

- Reduce the bandwidth of the system.
- Stiffen the machine so the load is not springy.

4.9.3 Non-Linear Mechanics

VECTOR**STAR** tuning is based on linear control theory. The most important requirement of a linear motor controller is that the total reflected inertia should not change substantially during operation. Load inertia includes all the inertia reflected to the motor, such as inertia through gearboxes and leadscrews. Inertia can change in ways that are easy to understand, such as the inertia of a spool of cable decreasing when the cable is unrolled. It can also change in less intuitive ways, such as chain drives (which have load in one direction but are unloaded in the other) and systems with excessive backlash (where there is no load when gear teeth are not touching).

When the inertia changes, the system has the following characteristics:

- System performance is excellent when the motor is in some positions and unacceptable when the motor is in other positions.
- Reducing the bandwidth eliminates the problem.

If the system performance is poor because of changing inertia, you can make the following corrections:

- Correct the system mechanics so that inertia is constant.
- Detune (reduce the bandwidth of) the system. If the times when your system will have excessively changing inertia are predictable, you can write your program to detune your system in these regions.

4.9.4 Resonance

Resonance is a high frequency (> 500 Hz) where the system mechanics oscillate. Normally, systems with resonance will be very stable when you tune with lower target bandwidths. As you increase the target bandwidth, you will begin to hear a fairly pure, high pitch. If you want to decrease resonance, use shorter, larger diameter driving shafts. Often, the low-pass filter can help you raise the bandwidth 20% or 30%, but this can be a difficult trial-and-error process: you slowly lower the low-pass filter frequency (LPFHZ) and attempt to raise the target bandwidth for tuning.

When your system has a resonance, it will have the following characteristics:

- The system will make a clear, high pitch (>500 Hz). Do not confuse this problem with compliance, which has a low pitch.

If the system performance is poor because of changing inertia, you can make the following corrections:

- Enable the low-pass filter (LPF) and reduce LPFHZ, if necessary.
- Reduce the bandwidth of the system.
- Shorten the length and increase the diameter of shafts and lead screws.

4.9.5 Low-Pass Filters

The LPF switch enables the low-pass filter. It can be turned on and off when the drive is operating. The frequency of the low-pass filter is stored in LPFHZ in Hz. It can also be changed when the drive is operating. For example, if LPFHZ is 200 and LPF is on, then a 200 Hz low-pass filter is run in the VECTOR**STAR**. The filter can be modeled as two cascaded, low-pass, single-pole filters, both with a 3 dB frequency of 200 Hz. LPFHZ should be set as high as possible, since it degrades the system performance.

For example, the following sequence sets the low-pass filter to 250 Hz and enables the drive.

```
LPF  ON       ;ENABLE LOW-PASS FILTER
LPFHZ  250    ;SET BREAK FREQ. TO 250
              ;HZ
```

If the low-pass filter is on, the TUNE command may not work well.

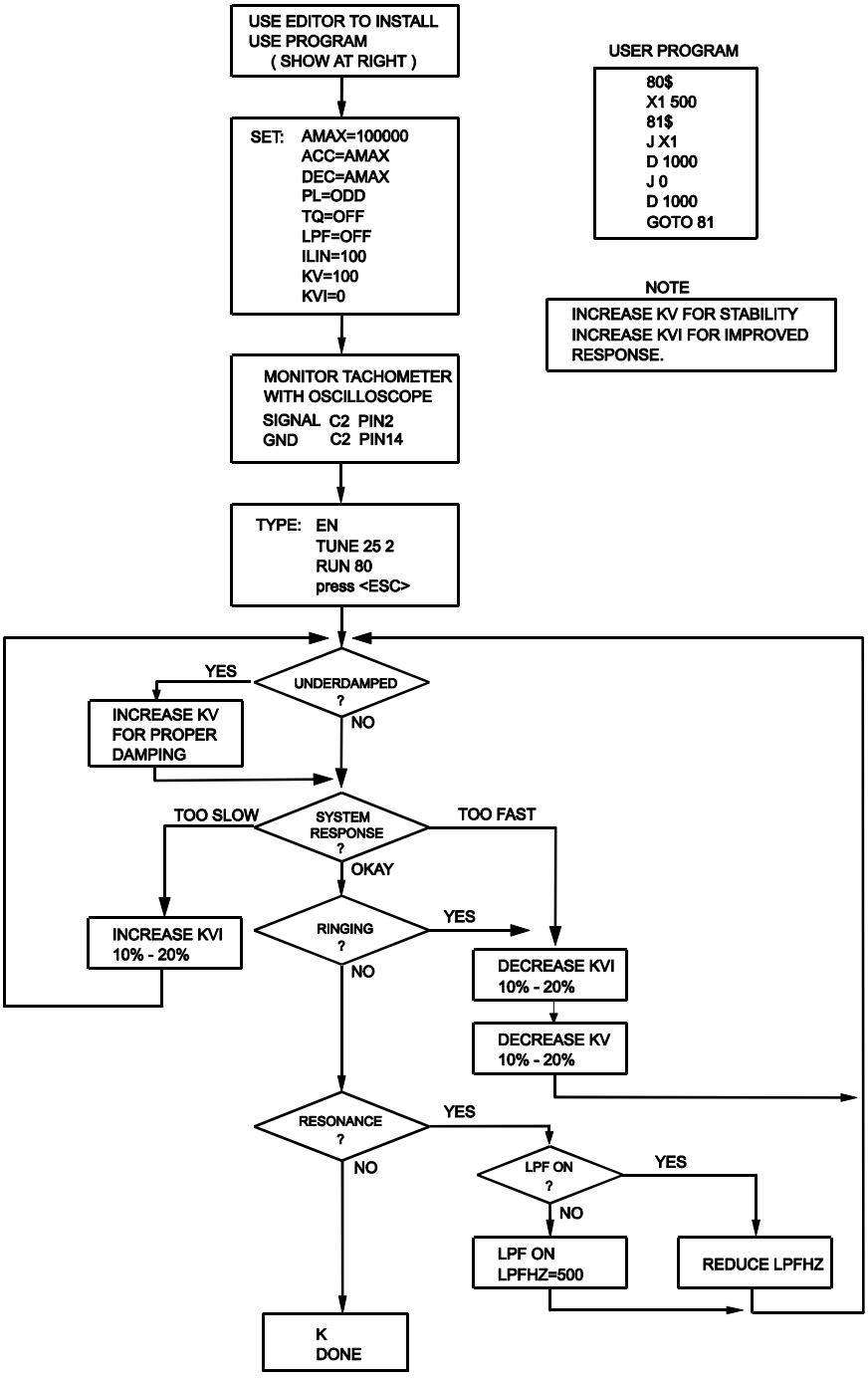


Figure 4.5 Velocity Loop Tuning flow Chart

CHAPTER 5
MAINTENANCE

5.1 INTRODUCTION

The information in this chapter will enable you to maintain the system's components, ensuring smooth, efficient operation of the motor. Preventative maintenance of the equipment is also specified along with periodic maintenance. Follow these practices when operating your servo system.

5.2 PREVENTATIVE MAINTENANCE

Preventative maintenance can help you prevent situations that will damage your equipment. Four types of preventative maintenance are presented below. Following each of the procedures can reduce problems with and add life to your equipment.

Preventative maintenance to this equipment must be performed by qualified personnel familiar with the construction, operation, and hazards involved with the application.

Electronic components in this amplifier are static sensitive. Use proper procedures when handling component boards.

Preventative maintenance should be performed with the VECTOR**STAR** system out of operation and disconnected from all sources of power.

5.2.1 Transient Voltages

All transient-producing devices must be properly suppressed.

Solid state controls of the VECTOR**STAR** may be affected by transient voltages. These voltages are in excess of the specified voltage for any given circuit. When these peak voltages occur, even for less than a second, permanent damage to the VECTOR**STAR** can occur.

In order to help avoid transient voltages that may interfere with electronic circuit functions within the PA and VECTOR**STAR**, all switched inductive devices or their wiring (solenoids, relay coils, starter coils, etc.) must be suppressed. A 220 ohm, 1/2 watt resistor in series with a 0.5 micro farad, 600 volt capacitor or equivalent is suggested.

5.2.2 Surge Current

Excessive current greater than that of the specified limits of the PA and VECTOR**STAR** can cause permanent damage to the system. Current limiting means are recommended to protect from these currents.

If the short circuit inrush current generated by the power source is in excess of 5000 amps RMS symmetrical current, an isolation transformer or line inductor must be utilized in the incoming power circuit. Failure to observe this precaution could result in damage to, or destruction of the PA and VECTORSTAR.

Input transformers step up or step down input voltage and can be either autotransformers or isolation transformers. Isolation transformers help eliminate the following:

- Damaging AC line voltage transients reaching the PA and VECTOR**STAR**.
- Damaging currents which may develop if a point inside the PA or VECTOR**STAR** becomes grounded.

5.2.3 Electrical Noise

The low levels of energy in the VECTOR**STAR** control circuits may cause them to be vulnerable to electrical noise. Sources of electrical noise are those pieces of equipment that have large, fast changing voltages and currents when they switch on and off. These devices have the capability of inducing critical current and voltage transients on their respective power lines. These transients must be accommodated for with noise immunity provisions.

Electrical noise is prevented with the same methods as Surge Current and Transient Voltages. However, there are other methods of preventing electrical noise. Such as:

- Maintain physical separation between electrical noise sources and the VECTOR**STAR** amplifier.
- Maintain physical separation between electrical noise sources and the VECTOR**STAR** control wiring. This can be accomplished by using separate conduits or wiring trays for control wiring and power wiring.
- Use twisted-pair wiring for control circuits of the VECTOR**STAR**.
- Follow good grounding practices when wiring the PA and VECTOR**STAR**. Be careful not to create a grounding loop with multiple ground paths. Follow the NEC's provisions on grounding.

5.2.4 Radio Frequency Energy

This equipment generates radio frequency energy.

This equipment uses, and can radiate radio frequency energy and must be installed and used in accordance with this installation and service manual in order to prevent possible interference with radio communications or other electronic equipment.

5.3 PERIODIC MAINTENANCE

Periodically you will need to inspect your equipment for possible problems to ensure ongoing safe and efficient operation. Periodic maintenance should be performed at scheduled intervals to insure proper equipment performance. It must be performed by qualified personnel familiar with the construction, operation, and hazards involved with the VECTOR**STAR** and its application. Power should be disconnected during all maintenance procedures.

5.3.1 Ventilation

The PA and VECTOR**STAR** should be mounted vertically to allow maximum ventilation of the components. This configuration allows the heat generated by the components to vent through the top and draft in cooler air through the bottom. The top and bottom of the components are vented to allow this drafting to occur. These ventilation passages should be kept open. If the PA requires auxiliary cooling with fans, inspect the fans on a regular basis.

5.3.2 Grounding Integrity

The method employed for grounding or insulating the equipment from ground should be checked to assure its integrity on a regular basis. This check should be performed with the power off and the testing equipment grounded.

CHAPTER 6
TROUBLESHOOTING

6.1 INTRODUCTION

The information in this chapter will enable you to order spare parts and isolate and resolve common system hardware problems. The VECTOR**STAR** aids in diagnostic evaluation through its LED Status Indicators and the VECTOR**STAR** Error Log. Both of these features are explained to assist you in finding solutions. As another part of Kollmorgen's obligation to its customers, Factory Support and Repair is also defined.

6.2 SPARE PARTS

There are no user serviceable parts on the VECTOR**STAR**.

There are several fuses that are user serviceable on the PA. Remember, the PA can be damaged by ESD (Electro-static discharge). Observe proper ESD protection practices.

The PA can be damaged by ESD (electro-static discharge). Observe proper ESD protection practices.

6.2.1 VECTORSTAR Spare Parts List

Connector Kit for 20 Amp VECTOR**STAR**s (VECTOR**STAR**-x20): VECTOR**STAR**C-101

6.2.2 PA Spare Parts List

Connector Kit:
PSR4C-100

Regen Fuses:

8.8 Ohms External Regen, 230 Volt Units (PA-2xx-xx01)
12 Amp Fuse: A-80552-002

6.2.3 Ordering Information

If you need to order parts for the VECTOR**STAR** and/or PA, you can order them through your local distributor. For a complete list of Kollmorgen representatives, contact us directly at:

Kollmorgen Motion Technologies Group
201 Rock Road
Radford, VA 24141
U.S.A.

Telephone: 540/639-2495
540/731-0847 (FAX)
540/875-3743 (TWX)

6.3 LED STATUS INDICATORS

6.3.1 VECTORSTAR LED's

The VECTOR**STAR** provides LED's for diagnostics. These LED's are on the front panel of the VECTOR**STAR**. When the VECTOR**STAR** is powered up, all LED's on the front panel turn on to verify they are operational. The states (on or off) of each LED are listed below.

- ACTIVE

This LED shows whether the VECTOR**STAR** is active. "Active" means that the VECTOR**STAR** is enabled and the REMOTE input switch is on. This LED is on when the VECTOR**STAR** is active and off when it is not active.

- SYS OK

This LED indicates the state of the hardware watchdog protection circuit. It should be on during normal operation. However, it is off during autobauding. If SYS OK turns off, take the VECTOR**STAR** out of service and contact the factory.

- CPU

This LED indicates the state of the software watchdog protection circuit. It is on during normal operation and blinks for the most serious errors. This LED also blinks during autobauding.

- FAULT

This LED indicates that a fault has occurred. A fault is an error that is serious enough to disable the VECTOR**STAR**. You can turn the FAULT LED on or off from your program. The FAULT LED is turned off when you enable the VECTOR**STAR**.

- RELAY

This LED indicates the status of the VECTOR**STAR** relay. It is on when the relay contacts are closed and off otherwise.

6.3.2 PA-Series LED's

The PA-Series power supply has 3 indication LED's, except low-current versions, which have 2:

- D.C. BUS

This green LED is on when AC Line Voltage is applied.

- REGEN

This yellow LED turns on when the PA regen circuit is active.

- OVERLOAD

This red LED indicates a fault. It turns on when the PA circuitry detects that the regen resistor has absorbed too much energy. It is turned off when power is removed and then reapplied. Normally, this is caused when the motor decelerates too rapidly or too often. If you get this fault, you may need to increase the power rating of your regen resistor and the PA may need to be modified at the factory. If your system has an internal regen resistor, you will need a new PA power supply designed for external regen. Contact the factory.

- BLOWN FUSE

This red LED indicates a fault. It turns on when the fuse that protects the regen resistor has absorbed too much energy. You must remove power and replace the fuse. The spare parts list at the beginning of this chapter provides ordering information for this fuse. If the fuse blows during normal operation, see the section above on "OVERLOAD" because similar conditions cause both faults and similar actions must be taken to correct the conditions. If you replace the fuse and it blows within a few seconds of applying power, then the regen transistor is probably shorted. The PA must be returned to the factory for repair.

The PA-X50 and PA-X75 (50 and 75 Amp versions) have 3 indication LED's:

- D.C. BUS

This green LED is on when AC Line Voltage is applied.

- REGEN

This yellow LED turns on when the PA regen circuit is active.

- FAULT

This red LED indicates an overtemperature fault. It turns on when the PA thermostat opens. It turns off when the thermostat closes. If this fault occurs, it means that the regen resistor is on too often or for too long. If you get this fault, you may need to increase the power rating of your regen resistor and the PA may need to be modified at the factory. Contact the factory.

6.4 ERROR LOG

The VECTOR**STAR** responds to a variety of conditions, both internal and external, hardware and software, which are grouped in a single broad category: errors. An error indicates that there is a problem somewhere. More serious errors are grouped as faults.

6.4.1 Error Levels

The VECTOR**STAR**'s response to an error depends on the error's severity. There are four levels of severity, listed below in increasing order:

Table 6.1 Error Severity Levels and Actions

1.	Errors that cause warnings.
2.	Errors that cause a program break and stop motion, in addition to Level 1 Actions.
3.	Errors that disable the system and set the FAULT LED, in addition to Level 2 Actions.
4.	Errors that disable almost all VECTOR-**STAR** functions (including communications) and flash the CPU LED to indicate the error number. These are called firmware errors.

When any error except a firmware error occurs, a message is displayed on the screen. The following items are printed: the error number, the offending entry, and an abbreviated error message. For example, disable the drive and type in a jog:

```
DIS
J 100
```

The VECTOR**STAR** will respond with:

```
ERR 50 'J 100'      VECTORSTAR
                    INHIBITED
```

The error number (50), the offending entry (the whole line), and the error message (you cannot command a jog when the drive is inhibited) are given on one 80-character line. The error message starts at character 40 so that if a 40-character display is used, the error message will not be printed. You can display the line directly, either with the Motion Link editor (GOTO A LINE NUMBER selection or ^Q^I), or with the VECTOR**STAR** Editor (P command).

Sometimes only an entry is bad and not the whole line. In this case only the bad entry is printed. For example,

```
PROP 2
```

generates:

```
ERR 83 '2' ;BAD OR OUT OF RANGE
```

since PROP is a switch and cannot be set to 2. If the error comes from the program, the line number of the offending entry is also printed. Use the Editor to enter these lines at the top of the user program:

```
11$
PROP 2
B
```

Exit the Editor and type:

```
RUN 11
```

and the response should be:

```
ERR 83 LINE 2 '2'   ;BAD OR OUT OF
                    ;RANGE
```

This message shows that the error occurred on line 2. You can enter the Editor and type:

```
P 2
```

and the following line will be displayed:

```
PROP 2
```

6.4.2 DEP

If your VECTOR**STAR** prints to a Data Entry Panel (DEP-01) or any other 40 character wide display, the standard error messages will not print properly. The problem is that error messages are based on an 80 character wide display and the DEP-01 is only 40 characters wide. To correct this problem, the VECTOR**STAR** provides the DEP switch, which, when turned on, cuts all error messages down to 40 characters. If your VECTOR**STAR** prints to a DEP-01, type:

```
DEP ON
```

6.4.3 Error History

The VECTOR**STAR** stores the twenty most recent errors in the Error History. To display the entire Error History, type:

```
ERR HIST
```

This causes the Error History to be sent to the terminal, with the most recent error sent first. When the VECTOR**STAR** is powered up, a "DRIVE POWERED UP" message is inserted into Error History even though this is not an actual error.

To clear the Error History, type:

```
ERR CLR
```

Error History remains intact even through power-down.

6.4.4 Displaying Error Messages

The ERR command can also be used to display an abbreviated description of the error. For example, type:

```
ERR 50
```

The VECTOR**STAR** responds with:

```
ERR 50 VECTORSTAR INHIBITED
```

You may display messages for errors from 1 through 999. If you type in an error number that the VECTOR**STAR** does not recognize, it will respond with:

```
ERROR NOT FOUND.
```

A description of all errors is given in Appendix C.

6.4.5 Firmware Errors

Firmware errors are an indication of a serious problem with the VECTOR**STAR**. These errors stop communications, disable the drive, and flash the CPU LED. The CPU LED flashes several times, then turns off and pauses. The number of flashes represents the error number. These error numbers range from 2 to 9. See Appendix C for information on these errors. Contact the factory should one of these errors occur. For a detailed discussion of firmware errors, see Chapter 8, Section 8.6

6.5 FACTORY SUPPORT AND REPAIR POLICIES

Kollmorgen is committed to helping you install, operate, maintain and troubleshoot your VECTOR**STAR** servo system. If your system did not pass the "Initial Check Out" tests or is not operating properly, then contact the Kollmorgen Field Service Department. Please see Chapters 7-10 before calling about software or programming questions. Be prepared to provide the full VECTOR**STAR** and PA model numbers listed on the front of your equipment. Contact us at:

Kollmorgen Motion Technologies Group
ATTN: Field Service Dept.
201 ROCK ROAD
RADFORD, VA 24141

TELEPHONE: 540/639-2495
540/731-0847 (FAX)
710/875-3743 (TWX)

CHAPTER 7
SOFTWARE INSTALLATION

7.1 INTRODUCTION

The VECTOR**STAR** comes with its own PC software interface called MotionLink Plus (which we will refer to by its generic name, MotionLink). MotionLink is a Windows-based program designed to most effectively assist the user in setting up and controlling the drive's operation. It is highly intuitive in nature and contains an extensive context-sensitive on-line Help tool (press F1 to access). The on-line Help describes how to use the program in detail and serves as a valuable reference for the drive's variable and command set, setup process, and troubleshooting techniques

Since the software is designed to guide the customer through the operation process of the drive, the MotionLink discussion in this section will cover only the installation of the program on the user's computer. Once the program is installed, the user can then start the program and press the F1 key for extensive, detailed help.

A "dumb terminal" can also be used to communicate to the drive. MotionLink also provides a terminal emulation feature that contains many useful editing tools for this type of communication. The following is recommended:

- When setting the values of drive variables, use the many graphical screens in MotionLink.
- When editing a program, use MotionLink's Program Editor screen.
- When running and debugging your program, use MotionLink's program Editor screen and the Terminal screen.

This chapter discusses many Processor Modes and Programming Modes, and much of the discussion revolves around how the different modes appear in the terminal interface. This is done so for clarity. The user may choose to operate in MotionLink's graphical screens, or from its terminal interface, whichever is more appropriate and more comfortable.

7.2 COMPUTER REQUIREMENTS

MotionLink requires an IBM-PC or compatible computer with the following features:

- IBM-PC, XT, AT, 386, 486, PS/2, or compatible computer running Windows 3.1 or Windows 95/98.
- 3.5" Floppy Drive.
- Standard Video Adapter (CGA, MDA, EGA, MCGA, and VGA).
- Serial Port (for communication link with SERVO**STAR**). The serial communications port may be COM1, COM2, COM3, or COM4. COM1 is the normal configuration:

COM1: Address 3F8h, Interrupt Request #4
COM2: Address 2F8h, Interrupt Request #3
COM3: Address 3E8h, Interrupt Request #4
COM4: Address 2E8h, Interrupt Request #3

7.3 INSTALLING MOTIONLINK PLUS

- Insert the MotionLink Plus for Windows diskette #1 into a 3.5" floppy drive.
- From the desktop (Win 95/98), select Start |Run. For Windows 3.1, select File | Run.
- Type "A:setup.exe" and hit the Enter key.
- Once Setup Screen is displayed, click on "OK" to start installation.
- Select the destination directory (The user may install MotionLink Plus for Windows in any directory or drive, but the default that is presented during installation is highly recommended).
- Click on "OK" to begin installation.
- Click on "OK" to create a "KMTG Motion Suite" Program Group in Program Manager.

7.4 RUNNING MOTIONLINK PLUS

- Win 95/98: Select Start -> KMTG Motion Suite -> MotionLink Plus.
- Windows 3.1: double-click the MotionLink Plus icon in the KMTG Motion Suite program group.

7.5 ACCESSING ON-LINE HELP IN MOTIONLINK PLUS

Once you have started MotionLink, pressing the F1 key at any time will give you access to on-line context-sensitive help. It is recommended that you make extensive use of MotionLink's on-line help file. It explains the operation of the program, and how to get the most out of it, in minute detail.

7.6 PROCESSOR MODES

7.6.1 Prompts

The VECTOR**STAR** provides several modes of operation. Each mode is distinguished by a unique prompt, the short series of characters that the VECTOR**STAR** writes to the screen requesting input. For example, the interactive prompt is —>.

The VECTOR**STAR** is designed to receive commands from a terminal or a computer through a serial port. In order to support computer communications, the following conventions are observed:

Table 7.1 VECTORSTAR Rules for Prompts

1. Prompts are 3 or 4 characters long.
2. Prompts end with a greater than symbol (>).
3. Each mode has a unique prompt.
4. Once the VECTOR**STAR** displays a prompt, it stops transmitting until a new instruction and/or a carriage return is received.

The last rule ensures that there is never a question about which device is transmitting. If a ">" has been issued from the VECTOR**STAR**, then the VECTOR**STAR** will not transmit anything until a command has been entered. The only exception is if you program the VECTOR**STAR** to print a ">" from a PRINT or INPUT command. The VECTOR**STAR** will allow ">" in print statements, though this is considered a poor practice if you are using a computer to communicate with the VECTOR**STAR**.

Similarly, the VECTOR**STAR** will not accept input unless a ">" has been issued. The INPUT command is the only exception to this rule. This rule can be awkward if you are using the VECTOR**STAR** from a terminal; if an error occurs during the interactive or monitor modes after the ">" has been displayed, the VECTOR**STAR** will not print the error message until a carriage return or escape has been entered.

The prompt for each mode is listed below. The only exception is the Run mode. This mode does not have a prompt since input is not accepted from the serial port. Notice that the trace prompt does not end with ">." This is because the trace prompt does not indicate that the VECTOR**STAR** is waiting for input. If the VECTOR**STAR** is communicating within a multidrop communication line, then the prompt is modified to include a prefix which indicates the axis address. Table 7.2 shows the prompts in both the single and multidrop configurations. Note that the multidrop address is 65 (ASCII "A").

Table 7.2 VECTORSTAR Prompts

Mode	Non-multidrop (ADDR=0)	Multidrop (ADDR = 65)
Interactive	`-->`	`A->`
Monitor	`==>`	`A=>`
Single-step	`s-->`	`As->`
Trace	`t...`	`At..`
Load	l->	Al>

7.6.2 Mode Descriptions

The following section describes each of the modes of operation. Figure 7.1 is a diagram of each mode and how it interacts with the other modes.

7.6.2.1 Interactive Mode

The VECTOR**STAR** normally powers-up in the Interactive mode. This mode allows you to start programs, display and change variables, and enter motion commands for immediate execution. The interactive prompt (-->) is written to the screen, and the VECTOR**STAR** awaits a new command.

Refer to Figure 7.1. There are many ways to enter the Interactive mode. First, if the power-up label (POWER-UP$) is not present, the VECTOR**STAR** will power-up in the Interactive mode. The BREAK (B) command and errors that break program execution cause the VECTOR**STAR** to exit the Run mode and enter the Interactive mode.

7.6.2.2 Run Mode

The VECTOR**STAR** is normally in the Run mode when a program is executing. There is no prompt because input is not accepted from the terminal. The program is running; it can display errors and print to the terminal.

Refer to Figure 7.1. After autobauding, the Run mode is normally entered from either the Interactive mode, the RUN command, or from multi-tasking. If the power-up label (POWER-UP$) is present, the VECTOR**STAR** will start running your program at that label on power-up.

Errors can also cause the VECTOR**STAR** to change modes. Some errors are serious enough to cause the VECTOR**STAR** to break program execution. Usually, this has the identical effect of issuing a BREAK (B) command.

As an option, you can write an error handling routine beginning at label ERROR$. This routine should be short and should end with a BREAK (B) command. The error handler is intended for graceful error recovery. For example, you can set outputs or print a message. It is not intended to continue the program as if the error never occurred.

7.6.2.3 Monitor Mode

The VECTOR**STAR** Monitor mode is a unique mode for positioners. In this mode, the user program is running, but commands are accepted from the terminal for immediate execution. The Monitor mode allows you to display and change variables during program execution, including tuning variables.

You can print and modify any variable from the Monitor mode. The commands that are allowed from the Monitor mode are a subset of the commands allowed from the Interactive mode as shown in Table 7.3.

Table 7.3 Monitor Mode Commands

?	;	B	DIS	EN
ERR	K	MOTOR	P	PS
R	RS	S	ZPE	

In the Monitor mode, all print commands from the user program are suppressed, and the monitor prompt (==>) is displayed. Print commands typed in from the Monitor mode are executed immediately.

To enter the Monitor mode, press the escape key while a program is running. Pressing the escape key again will change modes back to the Run mode. STOP, BREAK, and KILL all return the VECTOR**STAR** to the Interactive mode.

7.6.2.4 Single-Step Mode

The Single-Step mode is provided for debugging, and it allows you to execute a program one step at a time. The single-step prompt (s->) is printed out, followed by the line that is about to be executed (the *next* command). Any command allowed from the terminal in the Monitor mode is also allowed from the terminal in the Single-Step mode. These commands allow you to probe the VECTOR**STAR** variables while debugging your program. If you press the enter key without a command entered, then the next command in the user program is executed. To stop the program, enter the *S*, *B*, or *K* command. To turn off the Single-Step mode and allow the program to execute normally, press the escape key twice (once to get into the Monitor mode and again to get into the Run mode), or type *SS OFF*.

Single-Step mode is enabled by turning SS on, either from the program, from the Interactive mode before running the program, or from the Monitor mode. After SS is on, the VECTOR**STAR** will enter the Single-Step mode when the user program is executed. SS can also be turned on and off from the program. This is useful if there are certain sections that you want to single step through. Turning SS off from the program returns the VECTOR**STAR** to the Run mode.

7.6.2.5 Trace Mode

The Trace mode is provided for debugging. When in Trace, the VECTOR**STAR** prints statements before they are executed. The trace prompt (t...) is printed out, followed by the line that is about to be executed, and the line is then executed. This process is repeated for each command.

The Trace mode is enabled by turning TRC on. When TRC is on, the VECTOR**STAR** will enter the Trace mode when the user program is executed. TRC can be turned on and off from the Interactive mode before executing the program or from the program itself. It can be turned on from the Monitor mode. Pressing the escape key from the Trace mode will cause the VECTOR**STAR** to exit the Monitor mode and turn TRC off. If both TRC and SS are on, then the VECTOR**STAR** will be in Single-Step mode.

7.6.2.6 Other Modes

Other modes shown in Figure 7.1 include the communication modes (Program Load, Program Dump, and System Dump). These modes are covered in later chapters.

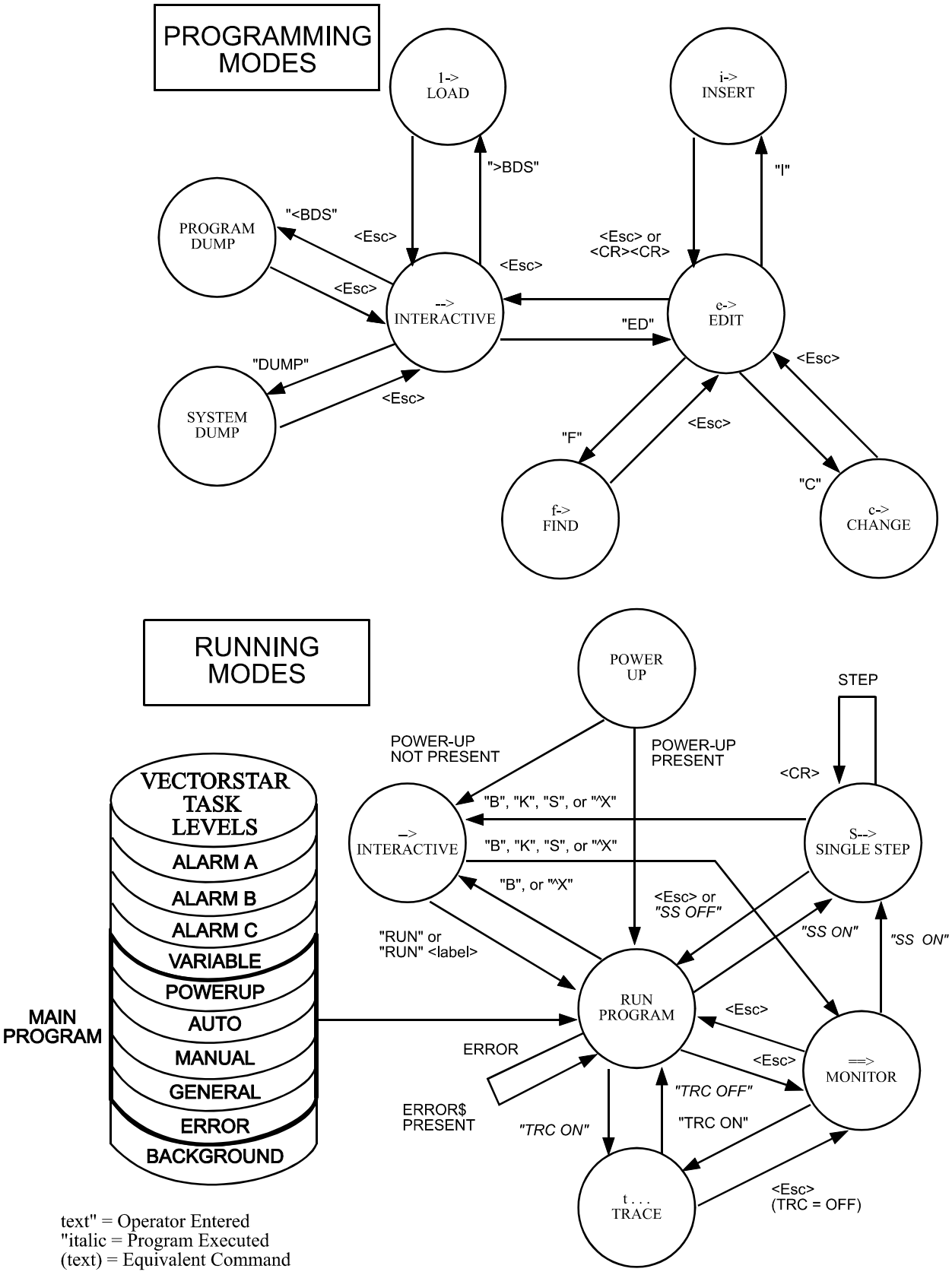


Figure 7.1 VECTORSTAR State Table

CHAPTER 8
GENERAL PROGRAMMING

8.1 INTRODUCTION

This chapter discusses the basics of the VECTOR**STAR** and its programming language.

Your VECTOR**STAR** system should be mounted and wired as described Chapter 2. The AC Line voltage to your PA should not be turned on for examples in this chapter until you are asked to do so. Turn on Control Power only and establish communications. If the proper connections are not made, or the terminal is not communicating, then see Chapter 2.


WARNING

AC LINE SHOULD NOT BE TURNED ON!

8.2 INSTRUCTIONS

The VECTOR**STAR** can respond to instructions entered from the terminal. The format of the instructions is usually a command followed by one or more parameters. For example, the jog instruction is a "J" followed by one parameter: the desired speed. The following example would cause the motor to jog at 10 RPM.

```
J 10
```

The command and parameter must be separated by at least one space.

8.2.1 Comments

Instructions can be followed by comments on the same line. A semicolon marks the beginning of a comment. The VECTOR**STAR** ignores everything on the line after the semicolon. For example:

```
J 10 ;THIS IS A GOOD COMMENT
```

is a valid instruction. Note that a space must separate the semicolon from the last parameter.

```
J 10;BAD COMMENT-";" MUST BE
;PRECEDED BY A SPACE

;GOOD LINE. SPACE NOT REQUIRED-
;WHOLE LINE IS A COMMENT
```

8.3 VARIABLES

The VECTOR**STAR** uses variables to monitor and control virtually all of its processes.

8.3.1 Variable Units

Some variables have implicit units associated with their values. For example, all variables that monitor or control velocity have velocity units. In addition there are acceleration units, current units, and position units. Appendix F lists each variable with its units. Units are programmable; when shipped from the factory the standard settings are as follows:

Table 8.1 Standard Units

Acceleration Units:	**RPM / Second**
Current Units:	% of Full Amplifier Output
Position Units:	Counts
Velocity Units:	RPM
External Position Units:	Counts*
External Velocity Units:	RPM*

* This assumes external source is a motor with the same resolution as the VECTOR**STAR**. That is, external velocity units are set the same as velocity units.

With standard units, position is expressed in resolver-to-digital (R/D) converter counts; if your VECTOR**STAR** is configured with the standard 12-bit resolution R/D converter, then one revolution is 4096 counts.

You can change the units to whatever is convenient for your application. For example, you can select Radians/Second instead of RPM. Also, units can be tailored to a specific machine. For example, if the VECTOR**STAR** is driving a lead screw, velocity could be programmed in inches/minute. If you want to change the units, see Chapter 9, "Machine Specific Units." Examples in this manual will assume that the VECTOR**STAR** is configured with standard units.

8.3.2 Three Types of Variables

The VECTOR**STAR** has many variables, all of which are listed in Appendix F. The variables can be divided into three groups: monitor, control, and user.

· MONITOR VARIABLES

Monitor variables watch the system. You may display their values or use them in calculations. However, as a rule, you may not change them. The VECTOR**STAR** automatically changes these variables to reflect its status. Position feedback, PFB, is an example of a monitor variable.

· CONTROL VARIABLES

Control variables allow you to change or limit some process in the VECTOR**STAR**. An example of a control variable is current limit, ILIM. ILIM limits the maximum current the VECTOR**STAR** can deliver. It can be changed at any time.

· USER VARIABLES

User variables allow you to store information for later use or hold intermediate results of calculations. They are discussed later in this chapter.

8.3.3 Variable Limits

All variables have limits. It is important to be aware of these limits, since attempting to set a variable to a value outside its limits generates an error. For example, ILIM must be between 0 and 100. The limits of each variable are listed in Appendix F.

8.3.4 Switches

Switches are variables that can be set to 0 or 1 only. In other words, they have limits 0 and 1. Aside from this restriction, this discussion about variables also applies to switches.

8.3.5 Printing Variables

All variables can be displayed. To display a variable on the terminal, you should use P, the PRINT command, or, if only one variable is to be printed, "P" can be omitted. For example, type:

```
P ILIM
```

or

```
ILIM
```

Since the standard setting of ILIM on most systems is 100, the terminal should display:

```
100
```

Suppose you want to display PFB, the position feedback. Type:

```
P PFB
```

The position feedback should now be displayed. Assuming the motor and resolver are connected to the

VECTOR**STAR**, rotate the motor shaft about half a revolution. Now, print PFB as above and notice that it has changed to reflect the new position.

8.3.6 Changing a Variable

Variables are changed with assignment instructions. An assignment instruction begins with the name of the particular variable, followed by "=" and ending with the new value. One or more spaces can be substituted for the "=". The following examples assign (or at least attempt to assign) ILIM a new value:

```
ILIM=10      ;CORRECT—ASSIGN A NEW
             ;VALUE TO ILIM
ILIM  10     ;CORRECT—THE '=' IS
             ;OPTIONAL
ILIM10       ;INCORRECT—THERE MUST
             ;BE A SPACE OR '='
```

A few systems are set up with ILIM less than 100. If your terminal displays a number less than 100, write it down for reference later in this chapter. The following examples will change ILIM, and it must be reset to its original value.

Type the following line on the terminal:

```
ILIM=10
```

Next, print the new value of ILIM with the P instruction:

```
P  ILIM
```

ILIM should now be 10. Return ILIM to its original value (normally 100) by typing:

```
ILIM=100
```

Print ILIM to make sure the change was carried out properly.

8.3.7 Programming Conditions

Most variables can be changed but some can be changed only under certain conditions. For example, the maximum acceleration level, AMAX, can be changed only when the VECTOR**STAR** is disabled. Attempting to change AMAX with the VECTOR**STAR** enabled will generate an error. The conditions under which a variable can be changed are called programming conditions. Some variables should never need to be changed after the VECTOR**STAR** has left the factory; these variables are called "factory settable." Attempting to change a factory settable variable will generate an error. The programming conditions of all variables are listed in Appendix F.

Limits and programming conditions for all variables are shown in Appendix F.

8.3.8 Power-up and Control Variables

Table 8.2 Power-Up State of Programmable Switches

OFF	ON	REMEMBER FROM LAST POWER-UP
CAP	CAPDIR	ABAUD
CLAMP	DIR	LPF
DEP	MULTI	XS1-XS50
EXTLOOP	PROMPT	PL
FAULT		PLIM
GATEMODE		ROTARY
GEAR		TRIP
O1 - O8		
PROP		
RAMP		
REG		
SS		
STATMODE		
TRC		
TQ		
WATCH		
ZERO		

Most control variables and all user variables are stored in non-volatile RAM; their values are not lost when the VECTOR**STAR** is powered down. In general, control variables are remembered, except the switches. Table 8.2 shows the state of all VECTOR**STAR** programmable switches on power-up. The output word, OUT, is set to zero shortly after power-up.

8.3.9 Initial Settings of Control and User Variables

This section briefly discusses the standard initial and power-up settings for all control and user variables. The learning process is simplified by using the standard settings which disable certain functions. Note that here, "initial" means "as shipped from the factory." However, initial does not imply factory settable; you can change values that are set initially at the factory but you cannot change factory settable variables.

A2D — Switch from analog input to the gear (encoder) input. Remembered through power-down.

ABAUD — Enable autobauding. Initially set to 1 and left at 1 for preliminary operation.

ACC — Acceleration rate, initially in RPM/Sec and set to 100000.

ADDR — Address for multidrop applications. Initially set to 0 for non-multidrop.

ADEN — Acceleration units denominator. Initially set to 1000 for RPM/Sec.

AMAX — Limits DEC and ACC, acceleration and deceleration rates. Initially in RPM/Sec and set to 100000.

ANUM — Acceleration units numerator. Initially set to 4474 for RPM/Sec.

BAUD — Baud rate for serial communications. Automatically set by autobaud. Normally, you do not need to set BAUD.

CAP — Enable position Capture mode. Set to 0 on power-up and normally left at zero for preliminary operation.

CAPDIR — Direction of position capture. Set to 1 on power-up. The value of this variable does not matter if CAP is 0.

CLAMP — Enables Clamp mode. Set to 0 on power-up and normally left at 0 for preliminary operation.

DEC — Deceleration rate, initially in RPM/Sec and initially set to 100000.

DIR — Sets VECTOR**STAR** direction. If 1, then positive motion is clockwise. If 0, then positive motion is counter-clockwise. This is set to 1 on power-up.

FAULT — Fault is automatically set and cleared by the VECTOR**STAR**. You can change its state during operation, though you do not need to change it during initial operation.

GATEMODE — Enable Gate mode. Set to 0 on power-up and normally left at zero for preliminary operation.

GEAR — Enable electronic gearbox. Set to 0 on power-up and normally left at 0 for preliminary operation.

GEARI — Number of teeth on the input "gear" for electronic gearbox. Initially set to 1. Value of this variable does not matter if GEAR is 0.

GEARO — Number of teeth on the output "gear" for electronic gearbox. Initially set to 3. Value of this variable does not matter if GEAR is 0.

IDEN — Current units denominator. Initially set to 100 for percent.

ILIM — Peak current limit. The initial value is listed on the Test and Limits (TL) sheet which should be enclosed with your system. Normally set to IMAX. However, you may want to reduce it for protection. The motor can normally run under no-load with 15-25% current, so you can set ILIM as low as 15 or 25 during preliminary operation.

INUM — Current units numerator. Initially set to 4095 for percent.

KC — Low speed "graininess" compensation. Almost always set to 200. See discussion in Chapter 4 where a procedure for fine-tuning this variable is given.

KF	Tuning gain for velocity feed-forward. Set to 0 for preliminary operation.
KP	Tuning gain for position loop. Leave at initial setting for preliminary operation. The initial value is listed on the Test and Limits (TL) sheet, which should be enclosed with your system. Use TUNE command to change if necessary.
KPROP	Tuning gain for proportional velocity loop. Leave at initial setting for preliminary operation. The initial value is listed on the Test and Limits (TL) sheet, which should be enclosed with your system. Use TUNE command to change if necessary.
KV	Tuning gain #1 for integrating velocity loop (when PDF=1). Leave at initial setting for preliminary operation. The initial value is listed on the Test and Limits (TL) sheet, which should be enclosed with your system. Use TUNE command to change if necessary.
KVI	Tuning gain #2 for integrating velocity loop. Leave at initial setting for preliminary operation. The initial value is listed on the Test and Limits (TL) sheet, which should be enclosed with your system. Use TUNE command to change if necessary. If PI Loop is used (PDF = 0), KVI is integrating Gain but with a different value than that for PDF = 1.
LPF	Enables low pass filter. The low pass filter is often required to reduce noise or torsional resonance. Leave at initial setting for preliminary operation. Set to 1 if system is too noisy. The initial value is listed on the Test and Limits (TL) sheet, which should be enclosed with your system.
LPFHZ	Low pass filter break frequency. The low pass filter is often required to reduce noise or torsional resonance. Leave at initial setting for preliminary operation. Reduce value if system is too noisy. The initial value is listed on the Test and Limits (TL) sheet, which should be enclosed with your system.
MULTI	Enable multi-tasking.
O1-8	General purpose outputs. Reset to 0 on power-up. These variables are discussed later in this chapter.
PDEN	Position units denominator. Initially set to 1.
PDF	1 for PDF control, same as BDS5, 0 for PI control, same as VFS5, special for a spindle. When PDF = 0, autotune is not available.
PECLAMP	Position error limit for clamping. Initially set to 100. Value of this variable does not matter if CLAMP is 0 during preliminary operation.
PEMAX	Position error limit for system. This variable is initially set to 32767 (its upper limit) for preliminary operation and can be reduced later.
PEXT	PEXT monitors the position of the external (master) axis. Initially this variable is undefined. Value of this variable does not matter during initial operation.
PL	Enable position loop. This variable is remembered.
PLIM	Enable software travel limits. This variable is remembered.
PMAX	Positive software travel limit. Initially set to 100. If PLIM is 0, the value of this variable does not matter.
PMIN	Negative software travel limit. Initially set to -100. If PLIM is 0, the value of this variable does not matter.
PROMPT	Set to 1 on power-up and almost always left at 1. When set to 0, all prompts (such as —>) which are normally sent to the screen are not printed. This allows you to print customized messages.
PNUM	Position units numerator. Initially set to 1.
PROP	Enable proportional velocity loop. This

	variable is set to 0 on power-up and usually left at 0 for preliminary operation.
PXDEN	External position units denominator. Initially set to 1. Value of this variable does not matter during initial operation.
PXNUM	External position units numerator. Initially set to 1. Value of this variable does not matter during initial operation.
REG	Enable Profile Regulation mode. Set to 0 on power-up and normally left at zero for preliminary operation.
REGKHZ	Profile regulation frequency. Initially set to 1000. Value of this variable does not matter if REG is 0.
SCRV	Set S-curve level. Initially set to 2.
SS	Enable Single-Step mode. Set to 0 on power-up and normally left at 0 for preliminary operation.
STATMODE	Set mode of STATUS output. Set to 0 on power-up and normally left at 0 for preliminary operation.
TMR1	Software timer. Set to 0 on power-up. Value of this variable does not matter during preliminary operation.
TMR2	Software timer. Set to 0 on power-up. Value of this variable does not matter during preliminary operation.
TMR3	Software timer. Set to 0 on power-up. Value of this variable does not matter during preliminary operation.
TMR4	Software timer. Set to 0 on power-up. Value of this variable does not matter during preliminary operation.
TRC	Enable Trace mode for debugging. Set to 0 on power-up and normally left at 0 for preliminary operation.
TRIP	Enable position trip points.
TQ	Enable torque loop, which disables velocity loop. This variable is set to 0 on power-up and left at 0 for preliminary operation.
VDEN	Velocity units denominator. Initially set to 10 for RPM.
VDEFAULT	Default velocity for MI and MA commands. Initially set to 1 RPM.
VNUM	Velocity units numerator. Initially set to 44739 for RPM.
VOFF	Offset velocity for electronic gearbox. Reset to 0 whenever GEAR is turned on. This variable should be left at 0 for preliminary operation.
VOSPD	Overspeed setting. Initially set to 1.2 times the VMAX value. This variable should be left at this value for preliminary operation, but it can be reduced for protection.
VXDEN	External velocity units denominator. Initially set to VDEN. Value of this variable does not matter during preliminary operation.
VXNUM	External velocity units numerator. Initially set to VNUM. Value of this variable does not matter during preliminary operation.
WATCH	Enable the serial watchdog timer. This function disables the VECTOR**STAR** if a command is not received from the serial port every WTIME milliseconds. Set to 0 on power-up.
WTIME	See WATCH above. Initially set to 1000.
X1..X2000	User variables. Initially set to 0.
XS1..XS50	User switches. Initially set to 0.
ZERO	Puts the VECTOR**STAR** in Resolver Zeroing mode. This is set to 0 on power-up. Zeroing mode is used only during installation. If 1, VECTOR**STAR** rotates the motor to the zero position. If 0, the VECTOR**STAR** controls the motor normally.
Z0	Dead band for the analog input, 1 unit will set dead band of 620 microvolts.

8.3.10 User Variables

User variables are like memory on a hand-held calculator. They can be used as application-specific variables or for storing intermediate results of complex calculations. There are 2000 user variables: X1, X2, . . . X2000. They can be displayed and assigned new values like other variables. They can store numbers that range from -2^{31} (-2,147,473,648) to $2^{31}-1$ (2,147,473,647). For example, if you want to store PFB, the position feedback, at a particular time and use it later in a calculation, you can assign PFB to a user variable. Type the following line on the terminal:

```
X1=PFB
```

Now, without moving the motor, print X1 and PFB by typing:

```
P  X1  PFB
```

This print statement prints both X1 and PFB on one line and should show them to have approximately the same value. Note that when the motor is disabled, the position feedback can change slightly, so there may be a small difference in the values. Turn the motor about one-half of a revolution and repeat the print statement from above. Notice that X1 has remembered the old position feedback while PFB has changed. X1 will not change unless you assign it a new value.

8.3.10.1 Indirect User Variables

An advanced method of accessing the values stored in user variables is called *indirect*. With indirect user variables, the specified user variable "points" at another user variable. Indirect references to variables have the format: X(Xn) where n is between 1 and 250. The value stored in the variable Xn specifies the variable that X(Xn) refers to. This is best illustrated with an example.

Suppose you want to look at either X1 or X2 when X10 is either 1 or 2. Type this example:

```
X1=100
X2=1000
X10=1          ;USE X10 TO POINT TO X1
P  X(X10)      ;PRINT WHAT X10 POINTS
               ;AT
```

The VECTOR**STAR** responds:

```
100
```

since X(X10) = X1 = 100.

Now type:

```
X10=2          ;USE X10 TO POINT TO X2
P  X(X10)      ;PRINT WHAT X10 POINTS
               ;AT
```

The VECTOR**STAR** responds:

```
1000
```

since X(X10) is now X2, which equals 1000. So printing X10 indirect, X(X10), prints the user variable at which X10 points, not X10 itself.

Indirect user variables are often used to look up data in tables. For example, they are often used in teach programs—programs that remember a large number of positions taught by the operator. In this case, many user variables are used to remember positions, and one variable is used to point at the group. Use indirect references with caution since it is easy to make mistakes with them.

8.3.11 User Switches

User switches are similar to user variables, except that they can only take on values of 0 or 1. A user switch can be used in place of a user variable if you only need to store 0 or 1. An example of a good place for a user switch would be to store information for go/no-go decisions. This saves user variables for other places.

There are 50 user switches ranging from XS1 to XS50. For example, type:

```
XS33=1
P XS33
```

and the VECTOR**STAR** should respond by printing 1.

8.3.12 Special Constants

The examples above have used decimal numbers in most of the assignments. There are four special constants that make the VECTOR**STAR** easier to use: ON, OFF, Y, and N. ON is the same as 1 and OFF is the same as 0. Similarly, Y is 1 and N is 0. These constants are normally used for switches. Compare the two statements:

```
O1=1
O1 ON
```

Although both statements have the same effect, the second is easier to read (more intuitive). When you write programs, the use of ON and OFF, and Y and N can make the program easier to understand. Note, however, that the P command normally prints numbers, not ON, OFF, Y, or N. For example:

```
O1=ON
P OUT
```

will result in "1" being printed, not "ON." Another point to recognize is that the equal sign (=) is optional. The two statements

```
O1=ON
O1 ON
```

produce identical results. The program can be more readable if the equal sign is not used with Y, N, ON, and OFF.

8.4 MATH

8.4.1 Hexadecimal

The VECTOR**STAR** allows constants to be entered in hexadecimal, or hex. Hex is base 16 representation, which is often used when programming computers. VECTOR**STAR** hex constants begin with a number and are followed by an "h." For example: 16h, 0Fh and 0FFh are all hex numbers. Appendix G shows the hex conversion of 0 through 255. From the appendix, you can see that hex 25 is equal to decimal 37. The following two instructions have identical effects because 25 hex equals 37 decimal.

```
X9=37
X9=25H
```

Sometimes, the first digit of a hex number can be a letter. In this case, the number must be preceded with a zero. For example:

```
X9=FFH      ;ERROR-HEX NUMBER
            ;MUST BEGIN WITH A
            ;NUMBER

X9=0FFH     ;VALID STATEMENT
```

Hex is useful when trying to use general purpose inputs to control the user program. See later in this chapter for more information about applying these inputs.

8.4.2 Algebraic Functions

The VECTOR**STAR** provides four standard algebraic functions: multiplication, division, addition, and subtraction. The usual algebraic operators (*, /, +, -) are used. Standard algebraic hierarchy is observed: all multiplications and divisions are done before any additions or subtractions. Parentheses are provided to override this precedence. Type in the following examples:

```
P 1+2*3     ;THIS PRINTS 7, NOT 9—* IS
            ;DONE BEFORE +
P (1+2)*3   ;THIS PRINTS 9
```

Math expressions must obey the rules listed in Table 8.3.

Table 8.3 Rules for Math Expressions

1.	No spaces are allowed.
2.	Any valid variables can be used.
3.	Any valid constants can be used.
4.	Indirect user variables can be used.
5.	Any math operator can be used.
6.	Parentheses can be nested to 2 levels.
7.	Integer math is used for all operations.
8.	Expressions are evaluated left to right.

Valid math expressions can be substituted for numbers in most instructions. A few examples of math expressions in assignment instructions follow:

```
X1=500
X1=5*100
X1=5000/10
X1=(7+3)*(28+22)
```

All set X1 to 500. Furthermore, variables can be used in the following expression, which fills X3 with 600.

```
X1=20
X2=30
X3=X1*X2
```

All operations are done with integer math. Fractional results from division are rounded to the nearest integer. Also, expressions are evaluated from left to right. These two conditions can cause unexpected results. Consider the following expressions:

```
P 53/100*280  ;THIS PRINTS 280
P 280/100*53  ;THIS PRINTS 159
P 280*53/100  ;THIS PRINTS 148
```

Mathematically, these three expressions are equivalent; they calculate 53% of 280, which is exactly 148.4. However, with integer math, the first expression is evaluated as 280. This is because 53/100 is evaluated first. The result, 0.53, is rounded to the nearest integer, 1, which is multiplied by 280. Likewise, in the second expression, the 280/100 is evaluated as 3, which is multiplied by 53 to get the result 159. Only the third expression gives the expected result, 148. In this example, round-off error is minimized by performing the multiplication first.

8.4.3 Logical Functions: AND, OR

Two logical math functions, AND and OR, can also be used in math expressions. ANDing is indicated by "&" operator and ORing is indicated by "!" operator. When evaluating an expression, AND has the same level of precedence as multiplication, and OR has the same level as addition.

Like hex, logical math is often used when programming computers. With logical functions, two numbers are converted to binary representation and compared bit by bit. When the numbers are ORed, if either bit is set, the result bit is set. With ANDing, both bits must be set for the result to be set. Type in the following examples:

```
P 1!2         ;THIS IS 3
```

The VECTOR**STAR** responds 3, since

```
          00000001 (Binary 1)
   OR     00000010 (Binary 2)

          00000011 (Binary 3)
```

```
P 1&2         ;THIS IS 0
```

The VECTOR**STAR** responds: 0, since

```
          00000001 (Binary 1)
   AND    00000010 (Binary 2)

          00000000 (Binary 0)
```

Logical math is generally used with hex constants.

Logical math is also useful when trying to use general purpose inputs to control the user program.

8.5 GENERAL PURPOSE INPUT/ OUTPUT

The VECTOR**STAR** provides 16 general purpose inputs and 8 general purpose outputs. On power-up, all outputs are turned off. Inputs and outputs can both be referred to individually or collectively: I1, I2, . . . I16 represent the individual inputs, and O1, O2, . . . O8 represent the outputs. You can turn the third output on and the sixth off by typing:

```
O3  ON        ;TURN ON THE THIRD
              ;OUTPUT BIT
O6  OFF       ;TURN OFF THE SIXTH
              ;OUTPUT BIT
```

To display the fifth input, type:

```
P  I5
```

Either 1 or 0 will be displayed.

8.5.1 Whole Word I/O

Inputs and outputs can also be referred to collectively. In order to do this, the individual inputs or outputs are referenced as the bits of a digital word, hence the term Whole Word I/O. Whole Word references are especially useful when you are trying to set or clear many output bits at once. If you are unfamiliar with logical/binary math or you plan to use I/O one bit at a time, you may not be interested in Whole Word I/O. However, it can save space and execution time when properly used.

Whole Word I/O is done using the variables OUT and IN. OUT is an 8-bit digital word representing all of the outputs, with O1 as the least significant bit (LSB), and IN is a 16-bit digital word representing all of the inputs, with I1 as the LSB. Each bit has a value which depends on its position within the word. The value in OUT or IN is the sum of the values for each bit that is turned on. The value for each bit is listed in Table 8.4.

Table 8.4 Output 1-8 Decimal Values

Out Bits	O8	O7	O6	O5	04	03	02	01
Value	128	64	32	16	8	4	2	1

For example, if O8 and O4 are on and all other outputs are off, then:

OUT = 128 (value of O8) + 8 (value of O4)
= 136.

Many bits can be set or cleared with one instruction. For example,

```
OUT=7
```

turns on O1, O2, and O3 while turning all other outputs off. One logical math statement can be used to set some bits without affecting others. For example:

```
O1 ON
O2 ON
O3 ON
```

can be replaced with:

```
OUT=OUT!7  ;SET 3 BITS WITH LOGICAL
           ;OR
```

which turns on O1, O2, and O3 without affecting O4 - O8. The logical AND can be used to turn off several bits:

```
OUT=OUT&7  ;CLEAR 5 BITS WITH
           ;LOGICAL AND
```

turns off O4-O8 and does not affect O1-O3.
Note that the hex representation can be especially useful when setting the higher bits:

```
O4 ON
O7 ON
O8 ON
```

is the same as:

```
OUT=OUT!0C8H
```

IN is formed with I1-16 in the same way OUT is formed with O1-8:

Table 8.5 Input 1-16 Decimal Values

In Bits	I16	I15	I14	I13
Value	32768	16384	8192	4096
In Bits	I12	I11	I10	I9
Value	2048	1024	512	256
In Bits	I8	I7	I6	I5
Value	128	64	32	16
In Bits	I4	I3	I2	I1
Value	8	4	2	1

For example, if IN were equal to 5010, that would mean I2, I5, I8, I9, I10, and I13 were on and all others were off, because 5010 is the sum of those bits:

5010 = 2 + 16 + 128 + 256 + 512 + 4096

8.6 ENABLE AND FAULT LOGIC

This section covers how to enable the VECTOR**STAR** and how faults affect the operation. This discussion will center around Figure 8.1. This drawing has six areas, each of which is labeled with a circled number, 1-6. Note that this drawing is a functional diagram; it does not directly represent the actual hardware and software used to implement these functions.

Your VECTOR**STAR** system should be mounted and wired as described in the Chapter 2. The AC Line to your PA should not be turned on for examples in this chapter until you are asked to do so. If the proper connections are not made, or the terminal is not communicating, see Chapter 2.


WARNING

AC LINE SHOULD NOT BE TURNED ON.

8.6.1 Firmware Faults, Area 1

Area 1 shows how firmware faults are combined. Firmware faults are the most serious errors. They include checksums (to help verify computer memory), watchdogs (to help verify that the computer is running properly), and the 5-volt logic power supply monitor.

These circuits are designed to watch the basic operation of the microprocessor. They do not generate error messages because the detected fault affects the microprocessor directly. Instead, they just blink the Central Processing Unit (CPU) LED.

As shown in Figure 8.1, firmware faults set a latch to turn off communications and blink the CPU LED. The CPU LED blinks in cycles consisting of 2 to 8 blinks and a pause. The number of blinks corresponds to the error number, which you can look up in Appendix C. The only way to reset these faults is to power-down the VECTOR**STAR**. These faults are serious and you should consult the factory if they occur. Do not confuse these faults with autobauding on power-up. When autobauding, the CPU LED blinks at a constant rate, about three times per second.

8.6.2 Fault Logic, Area 2

The large OR gate in Area 2 combines three types of faults: hardware, software, and firmware. The circuits that generate these faults are typical of motor controllers and are listed on the drawing. These faults are errors that are serious enough to disable the VECTOR**STAR**, as described in Appendix C.

8.6.3 Fault Latch, Area 3

The latch in Area 3 turns on the FAULT LED, the FAULT software switch, and the FAULT output on Connector C8. Any fault sets this latch; you can also write your program to turn it on if you detect a fault condition. The fault latch can be reset by:

1. Turning FAULT off,
2. Typing the enable command (EN), or
3. Powering down the VECTOR**STAR**.

8.6.4 Ready Latch, Area 4

Area 4 shows the logic required to make the drive ready. If there are no faults, the EN command sets the ready latch. This turns the READY software switch on. This latch is reset with the KILL (K) command, the DISABLE (DIS) command, or a fault. These turn READY off.

8.6.5 ACTIVE, Area 5

Area 5 shows that ACTIVE will be on if both READY and REMOTE are on. This turns on the ACTIVE LED. It also allows the VECTOR**STAR** to actively control the motor.

REMOTE (Remote Enable) is an isolated input that is accessed from Connector C2 on the front of the drive. You can print REMOTE with the P command. It must be 1 to activate the VECTOR**STAR**. If you cannot turn REMOTE on, see the Chapter 2. Note that some faults "hide" the value of the REMOTE input from the VECTOR**STAR** microprocessor. This does not normally matter because all faults must be cleared before the drive will enable. If this condition exists, the VECTOR**STAR** will print REMOTE as "-1."

8.6.6 Relay and STATUS Control, Area 6

Area 6 shows how software switch STATUS and the relay work. You can configure STATUS to indicate either drive READY (but not necessarily ACTIVE) or drive ACTIVE. The difference is in how you want to use STATUS. For example, if STATUS is used for an interlock, you want STATUS to indicate drive ACTIVE. If the VECTOR**STAR** becomes inactive for any reason (including the REMOTE input turning off), then STATUS will turn off. As an alternative, you can use STATUS to indicate that the VECTOR**STAR** is ready for the REMOTE input to turn on. That is, if REMOTE turns on, the VECTOR**STAR** will be ACTIVE. In this case, you want STATUS to indicate drive READY.

The software switch STATMODE controls which state STATUS will indicate. If STATMODE is on, then STATUS will indicate drive READY. If STATMODE is off, then STATUS will indicate drive ACTIVE.

The operation of STATUS is shown by the AND-gate and OR-gate in Area 6. If STATMODE is on, then READY will turn on STATUS through the AND-gate. If STATMODE is off, then only ACTIVE (from Area 5) will turn on STATUS through the other leg of the OR-gate. The STATUS output on optional Connector C8, Pin 35, is always the same as the STATUS software switch. Note, however, that the state of the STATUS output is undefined for 25 milliseconds after power-up.

STATUS may turn on for up to 25 milliseconds during power-up.

8.6.7 Motor Brake

Kollmorgen motors can be purchased with an optional brake. The brake is fail-safe in that if no current is applied, the brake is active. If you set STATMODE to 0, you can use STATUS to control the brake. Then, when

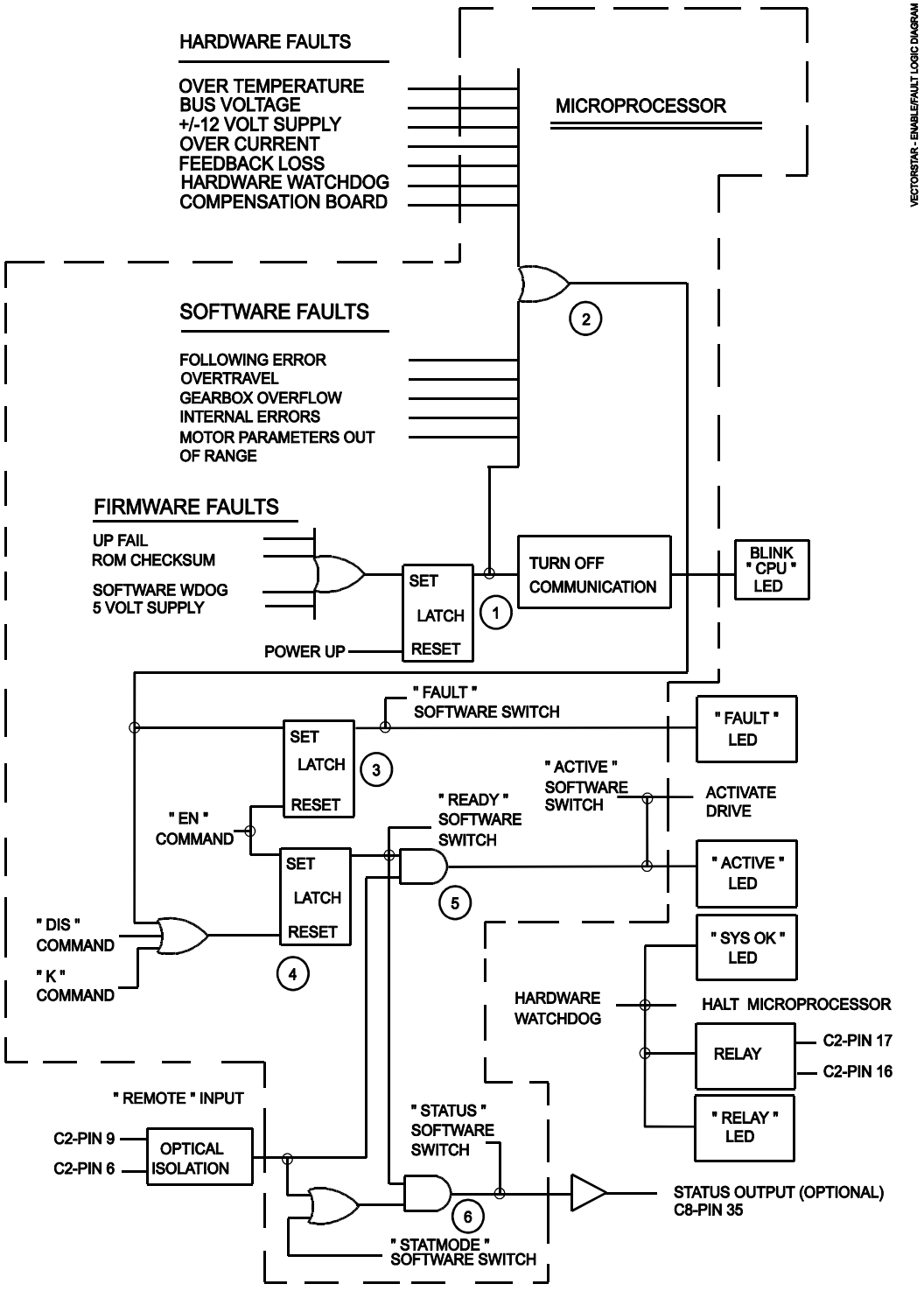


Figure 8.1 VECTORSTAR Enable/Fault Logic Diagram

the VECTOR**STAR** is disabled or powered down, the brake will be active.

8.6.8 Output Relay

The relay (Connector C2, Pins 16 and 17) represents the state of the hardware watchdog. The hardware watchdog makes a system more reliable because the watchdog is independent of the microprocessor. If the processor is not working, the watchdog will usually detect it (though this is not guaranteed).

On power-up, the contacts are open until the VECTOR**STAR** passes its power-up self tests. Then the contacts close and the VECTOR**STAR** begins normal operation. Note that if the VECTOR**STAR** is set to autobaud on power-up, the contacts will not close until after autobauding and establishing communications. One way to use the relay is to interconnect it with the main power contactor. In this case, a hardware watchdog fault will disconnect all power to the system.

The SYS OK LED indicates that there is not a hardware watchdog fault. If this LED goes out, you should remove the VECTOR**STAR** from operation and contact the factory.

8.7 DRIVE CONTROL

This section discusses several variables that you must be familiar with before you can control the VECTOR**STAR**.

8.7.1 Direction Control, DIR

DIR is a switch that controls the algebraic sign of command and feedback variables. When DIR is on, clockwise position, velocity, and torque are all positive. If DIR is off, then clockwise position, velocity, and torque are negative. DIR is turned on at power-up.

8.7.2 Position

8.7.2.1 Position Command and Feedback: PCMD & PFB

PCMD is the commanded position. It is generated internally from motion commands like the JOG command. PCMD is in position units. The standard position units are R/D converter counts as specified in Table 8.6. PCMD is set to PFB when the VECTOR**STAR** is disabled.

PFB, the position feedback, is the actual position of the motor. It is updated every millisecond. PFB is in position units. Section 8.3.5 (Printing Variables) explained how to look at PFB and watch it as the motor turns. PFB is always active, even when the VECTOR**STAR** is disabled. PFB is reset to zero when the VECTOR**STAR** is powered-up.

8.7.2.2 Position Error: PE & PEMAX

PE is position error, sometimes referred to as following error. It is the difference between PCMD and PFB. PE is zero when the VECTOR**STAR** is disabled. PE is in position units.

When the magnitude of the position error exceeds the value stored in PEMAX, a Position Error Overflow error is generated. This is a serious error, disabling the VECTOR**STAR** immediately. Note that setting PEMAX to some value will not limit the position error. The position error depends on the control loop parameters and the application. Normally, you want to set PEMAX to as low a level as will allow the system to run reliably. Setting PEMAX too low can generate nuisance errors since the position error has some variation during motion. PEMAX is in position units.

Position error is limited to protect the system. Excessive position error can indicate a fault condition. For instance, bearings wear out over the life of a motor. The increased load from worn bearings can increase the position error during motion. In many cases position error is the first indication of wear.

8.7.2.3 R/D Position, PRD

PRD is the output of the resolver-to-digital (R/D) converter in counts. PRD is not in position units. If your system has the standard 12-bit R/D converter, then 4096 counts will equal one revolution. The following table shows the R/D ranges versus resolution.

The VECTOR**STAR** should be disabled at this point (use the K or DIS command if it is enabled). PRD can be printed on the screen. From the terminal, type:

```
P PRD
```

and the R/D output will be displayed on the screen. Move the motor shaft by hand to several positions, printing PRD each time. Notice that PRD changes for each position. The VECTORSTAR supports 10-bit resolver resolution for high speed applications.

8.7.2.4 Sampling PFB, PCMD, and PEXT

When PFB and PCMD are used on the same line, they are always sampled during the same sampling interval (millisecond). This allows you to use PCMD, PFB, and a third variable called PEXT, which is discussed later in this chapter, without concern that the variables might be sampled at different times. For example:

```
P  PCMD "-" PFB " = " PCMD-PFB
```

This command would print the expected results, because the VECTOR**STAR** stores PCMD and PFB at the beginning of every command, then uses those stored values when the command is executed. On the other hand, if you type:

Table 8.6 PRD: Ranges and R/D Resolutions

R/D Resolution	PRD Min	PRD Max
12-Bit	0	4095
14-Bit	0	46383
16-Bit	0	65535

```
P  PCMD "-" PRD " = " PCMD-PRD
```

the results may not be as expected. This is because PRD is not stored at the beginning of the command. If the motor is turning, the two references to PRD will produce different results. This command takes up to 6 milliseconds to execute, and PRD can change several times while this command is executing.

8.7.3 Velocity

8.7.3.1 VCMD, VFB, VE, and VAVG

VCMD is the commanded velocity. Like PCMD, it is generated internally from motion commands. VCMD is zero when the VECTOR**STAR** is disabled. VCMD is in velocity units.

VFB is the feedback velocity. It is updated every millisecond. VFB is always active, even when the VECTOR**STAR** is disabled; if you turn the motor shaft by hand and print VFB on the terminal, you can see the velocity changing. Because VFB is updated very rapidly, the speed can appear to vary, even when the motor is rotating at a fairly constant speed. This is because the VFB shows the speed averaged over only 1 millisecond. The speed from one millisecond to the next normally varies a few RPM. The long term speed (measured over a few seconds) normally varies much less (about 0.01%). VFB is in velocity units.

VE is velocity error. VE is the difference between VCMD and VFB in velocity units.

VAVG is the average of VFB over the previous 16 milliseconds. Occasionally, the normal sample-to-sample variation of VFB is undesirable. In these cases, use VAVG.

8.7.3.2 Velocity Limits: VMAX and VOSPD

VMAX is the VECTOR**STAR** maximum velocity. It depends on the motor and the resolution of the R/D converter. For standard systems with 12-bit R/D converters, VMAX is less than or equal to 12000 RPM. For 14-bit systems, VMAX is limited to 3000 RPM; 16-bit systems are limited to 750 RPM. VMAX is set at the factory, and VMAX is in velocity units.

VOSPD is the maximum velocity for your system. The VECTOR**STAR** generates an overspeed fault if VFB is ever greater than VOSPD. You can set VOSPD to any level below 1.2 times the VMAX value. This allows you to limit the speed of your system to any level below VMAX. When an overspeed occurs, the VECTOR**STAR** is disabled immediately.

You should set VOSPD to at least 10% or 15% above your system's maximum speed to avoid nuisance overspeed faults. You can change VOSPD only when the VECTOR**STAR** is disabled. VOSPD is in velocity units.

8.7.4 Current

8.7.4.1 Motor Current: ICMD and IMON

ICMD is commanded motor current. ICMD, like PCMD and VCMD, is generated internally from motion commands. ICMD is in current units.

IMON is the output of the current monitor circuit, and it represents the magnitude of the motor current. IMON is always positive, and it is in current units. IMON is the digital conversion of the analog signal I_Monitor on Connector C2.

8.7.4.2 Current Limits: IMAX and ILIM

IMAX is the maximum level of current that the VECTOR**STAR** can output. It is set at the factory; its value depends on both the system rating and on the motor. IMAX is in current units.

ILIM limits the peak of ICMD, the commanded current. You can set ILIM to any level below IMAX. This allows you to limit the current below the maximum level that the VECTOR**STAR** can output. You can set ILIM at any time, even during profile moves. ILIM is in current units.

8.7.5 Enabling the Position Loop with PL

PL is a switch that controls the position loop. If PL is on, then the position loop is enabled. If PL is off, then it is disabled, and the VECTOR**STAR** is running as a velocity loop only. Most positioning applications run with PL on. See Section 8.9.1 for more information about the position loops. PL is remembered through power-down. You can change PL at any time.

8.7.6 Controlling the Velocity Loop with PROP

PROP is a switch that controls the integration section of the velocity loop. If PROP is on, then the velocity loop is proportional and the integral is disabled. If PROP is off, then the velocity loop is fully integrating. PROP is turned off at power-up. You can change PROP at any time. Most applications run with PROP off. Sometimes proportional velocity loops are used during set-up. See Section 8.9.2.1 for more information.

8.7.7 Enabling the VECTORSTAR


WARNING

THE VECTORSTAR WILL BE ENABLED AND THE MOTOR WILL TURN. SECURE THE MOTOR.

At this point you should turn REMOTE on as described in the Chapter 2. Type the following command to print the state of the REMOTE input:

```
P REMOTE   ;REMOTE SHOULD BE 1
```


WARNING

SHOCK HAZARD!

Large voltages from the AC Line and the DC Bus can cause injury. Ensure that the wiring is correct. See Chapter 2.


WARNING

THE MOTOR MAY MOVE UNEXPECTEDLY!

BE PREPARED TO DISABLE THE VECTORSTAR!

You should have completed "Initial Check-Out" in the Chapter 2. If not, complete that section before proceeding.

This section will enable the VECTORSTAR. The system may be unstable. The motor may begin oscillating or run away. Be prepared to disable the VECTORSTAR quickly. You can disable the VECTORSTAR by turning off (opening the contacts of) LIMIT or REMOTE.

To enable the VECTOR**STAR**, turn on the AC Line and enter the enable command:

```
EN
```

The VECTOR**STAR** should turn on. To verify that it did turn on, print ACTIVE. If ACTIVE is 1, then the VECTOR**STAR** is enabled; otherwise, it is disabled.

To disable the VECTOR**STAR**, enter the disable command:

```
DIS
```

As an alternative, you can disable the VECTOR**STAR** with the one-letter kill command by typing:

```
K
```

ENABLE, DISABLE, and KILL are examples of VECTOR**STAR** commands. All of the VECTOR**STAR** commands are listed, with their formats and syntax, in Appendix G.


NOTE

Appendix G is a quick reference for all VECTORSTAR commands.

8.7.8 Limiting Motor Current

The following section discusses how the VECTOR-**STAR** limits motor current.

8.7.8.1 Continuous Current, ICONT

The VECTOR**STAR** limits current in two ways: peak current is limited according to the variable ILIM, which was discussed earlier in this chapter; continuous (that is, average) current is limited according to the variable ICONT. The software that limits the time that motor current is allowed to be above ICONT is called *foldback,* since the current is gradually folded back to ICONT. ICONT is dependent on the VECTOR**STAR** rating and on the motor; ICONT is set at the factory, and it is in current units.

Most VECTOR**STAR** systems have about 2:1 peak to continuous rating. Generally, ILIM is 100% of the maximum current and ICONT is about 50%. The purpose of the foldback software is to allow the output current to go above ICONT for a short time (generally 2-3 seconds) while still protecting the VECTOR**STAR** from overheating.

8.7.8.2 Foldback Current, IFOLD

There are two current limits: ILIM and IFOLD. ICMD (the commanded current) is limited by either ILIM or IFOLD, whichever is less. You can set ILIM but you cannot set IFOLD; IFOLD is controlled by the foldback software. IFOLD depends on three things: ICONT (continuous current rating of the VECTOR**STAR**), IMON (current monitor), and time.

When the VECTOR**STAR** is disabled, IFOLD is set to some value well above maximum current (IMAX), and thus, well above ILIM. Since current is limited by the lesser of ILIM and IFOLD, IFOLD has no effect under this condition. If IMON, the output current, stays below ICONT, then IFOLD remains at its original, high value. If IMON is greater than ICONT, IFOLD gradually decreases. The greater IMON is, the faster IFOLD decreases. Since IFOLD starts out well above ILIM, initially this has no effect. However, when IFOLD is less than ILIM, IFOLD will limit the current. This is called "being in foldback." If IMON remains (on average) above ICONT long enough, IFOLD will decrease all the way to ICONT, forcing IMON eventually to become less than or equal to ICONT. Typically, it takes at least 2 to 3 seconds for IFOLD to decrease from its original high value to IMAX. At this point, the VECTOR**STAR** is in foldback. It takes an additional 10 seconds to reduce IFOLD from IMAX to ICONT.

If IMON is reduced below ICONT, then IFOLD will increase; the smaller IMON is, the faster IFOLD will increase. If IMON remains below ICONT long enough, IFOLD will return to its original high value.

8.7.8.3 Monitoring Current Limits

There are two switches that provide information on current limiting. SAT is a switch that is on if the current is limited by either ILIM or IFOLD. FOLD is a switch that is on if the current is limited by IFOLD only.

The operation of the foldback software is as follows:

If... IMON > ICONT then... IFOLD decreases
If... IMON < ICONT then... IFOLD increases

If... IFOLD < ILIM then... FOLD is on
If... IFOLD > ILIM then... FOLD is off

If... ICMD = ILIM or IFOLD then... SAT is on
If... ICMD < ILIM and IFOLD then... SAT is off

ICMD is never > ILIM
ICMD is never > IFOLD

In some cases, it may be desirable to know when foldback is just about to limit current below ILIM. You can use IFOLD for this; if IFOLD is less than ILIM, the foldback software is limiting current. If IFOLD is larger than ILIM, but only by 5% or 10%, then foldback software is about to limit current.

8.8 MOTION COMMANDS

This section discusses how to control motion using the VECTOR**STAR**. Basic motion commands are described first. Later sections discuss advanced motion control, including Macro Moves, electronic gearbox, and synchronizing motion.

8.8.1 Basic Motion Commands

8.8.1.1 AMAX, ACC, & DEC

The VECTOR**STAR** controls acceleration with three variables: AMAX, ACC, and DEC.

AMAX is the maximum acceleration allowed for almost all motion commands. The only exception is electronic gearbox. AMAX is the upper limit for the normal acceleration rates, ACC and DEC. AMAX should always be set below the acceleration level that can damage your machine. Errors that stop motion will

decelerate the motor at AMAX; therefore, your machine is subject to deceleration rates of AMAX at any time. AMAX is in acceleration units, which are RPM/second as a default. AMAX can be changed only when the VECTOR**STAR** is disabled.

Set AMAX below the maximum acceleration rate that your machine can experience without damage.

ACC is the acceleration rate for most moves. ACC is in acceleration units. ACC can be changed at any time, although it must be less than AMAX. Attempting to set ACC to a value greater than AMAX will generate an error.

DEC is the deceleration rate for most moves. DEC is also in acceleration units. DEC can be changed at any time. Attempting to set DEC to a value greater than AMAX will generate an error.

8.8.1.2 EN

Before any motion can take place, the VECTOR**STAR** must be enabled. Type:

```
EN
```

8.8.1.3 Enabling Motion with MOTION

MOTION is a hardware input that enables or inhibits motion. If MOTION is on, motion is enabled; if MOTION is off, motion is inhibited. You can enable the VECTOR**STAR** if MOTION is off, but commanding motion will generate an error. If you do not need to connect MOTION for your application, you must hardwire MOTION on. See Chapter 2 for instructions on how to hardwire MOTION. Before continuing, make sure that MOTION is on. Type the following command to print the state of the MOTION input:

```
P MOTION    ;MOTION SHOULD BE 1
```

Many times, the MOTION input is controlled by the normally-closed contacts of a push button. This push button is often called "STOP," since pressing the button opens the MOTION input and forces the motor to stop. Emergency Stop should not be implemented with the MOTION input. Emergency Stop should be connected to a contactor that removes power from the system. This is because an emergency stop, which is for safety, should not depend on VECTOR**STAR** functions to operate properly.

Do not use MOTION or any other VECTORSTAR input for Emergency Stop. When Emergency Stop is activated, it should directly remove power from the system.

8.8.1.4 STOP (S) Command

Any motion can be stopped using S, the STOP command. S has no parameters. S decelerates the motor at AMAX and terminates all motion commands. The S command does not disable the VECTOR**STAR**.

Normally, the STOP command should only be given from the terminal or from the program in response to an error condition. A better method for stopping motion from the program under normal circumstances is:

```
J 0    ;JOG TO 0 SPEED—STOP MOTION
       ;AT DEC, NOT AMAX
```

The *J 0* command also stops motion from any mode, much like the STOP command. Unlike *S*, *J 0* decelerates at the rate specified by DEC.

The *S* command should not be used as a part of normal program operation. Use *J 0*.

At any time, when motion is commanded, if the MOTION input turns off, an error is generated, and all motion is stopped, as if the STOP command were given. Also, any errors with a severity of 2 or 3 will stop motion in a straight line deceleration at a rate of AMAX. Appendix C lists all errors and their severity.

8.8.1.5 STOP and BREAK with Control X (^X)

You can execute a stop and break command with the control-X (^X) character. Control-X or ^X means that you hold down the control key (Ctrl) on your terminal (or IBM-PC) and press the X key. This has the same effect as typing B, then S from your terminal.

8.8.2 Limiting Motion

The VECTOR**STAR** allows you to limit motion of the motor with both Software and Hardware Travel Limits.

8.8.2.1 Hardware Travel Limits

If you have an application with boundaries which should never be crossed, you are encouraged to use the Hardware Travel Limits with limit switches.

Exceeding Hardware Travel Limits is a more severe error than exceeding Software Travel Limits. The VECTOR**STAR** assumes that Software Travel Limits should catch normal overtravel conditions and that a Hardware Travel Limit indicates a serious problem. Hardware Travel Limits disable the VECTOR**STAR** rather than just stopping motion, as the software limits do. This means that the motor must be backed away from the limit by hand.

Chapter 2 discusses how to wire LIMIT. Usually, two limit switches are wired in series and connected to LIMIT; the contacts of these switches must be closed for the VECTOR**STAR** to be enabled. If the contacts open, the VECTOR**STAR** will be disabled, the motor will coast to a stop, and an error will be generated. This limit is a safety device and not part of normal program operation. Hardware Travel Limits are always enabled.

8.8.2.2 Software Travel Limits: PMAX and PMIN

There are two software limits: maximum and minimum. If position feedback (PFB) moves outside the software limits, an error is generated and motion stops. Software Travel Limits are intended as a guard against motion that is out of range due to improper operation or programming errors.

PMAX is the maximum position allowed and PMIN is the minimum. If PFB is greater than PMAX, negative motion is allowed, but positive motion is not. If PFB is less than PMIN, only positive motion is allowed. PMAX and PMIN are in position units and can be changed at any time.

Software Travel Limits are enabled with PLIM, which can also be changed at any time . If PLIM is on, software limits are active; otherwise, PMIN and PMAX are ignored. PLIM is remembered. If you have an application with boundaries which should not be crossed, you are encouraged to use Software Travel Limits.

Note that you should set DIR before setting the Software Travel Limits. This is because DIR relates PMAX and PMIN to clockwise and counter-clockwise motion limits. If you change DIR, you must reset PMAX and PMIN.

8.8.2.3 User Position Trip Points: PTRIP1 & PTRIP2

The VECTOR**STAR** provides two user position trip points, which control a switch. You can use this switch to control your program.

The two trip points are PTRIP1 and PTRIP2. Both are in position units. You can program either at any time. If the position feedback (PFB) is greater than or equal to PTRIP1, then the TRIP1 switch will be on. If PFB is less than PTRIP1, then TRIP1 will be off. Similarly, if PFB is greater than or equal to PTRIP2, then TRIP2 will be on; otherwise, TRIP2 will be off.

Trip points are not limits in the sense that they do not inhibit motion. Trip points convert position feedback to an on-or-off signal. Trip points are particularly useful with alarms and the HOLD command, both of which are presented in Chapter 9 (User Programs).
Position trip points require a lot of calculations. As a result, they slow the execution of the user program by about 4%. If you are not using trip points, you can disable them by typing:

```
TRIP OFF
```

When the VECTOR**STAR** is powered-up, TRIP and trip points are remembered.

8.8.3 Profiles

When a positioner commands the motor to move from one point to another, it must control acceleration, deceleration, and traverse speed. The velocity of the motion versus time is called the profile. Simple profiles begin and end at zero speed and have three segments: acceleration, traverse, and deceleration. You must specify ACC, the acceleration rate, and DEC, the deceleration rate, before commanding the move. The traverse speed and the distance to move are specified in the move command itself.

The graph in Figure 8.2 shows a simple profile. The move begins at position 0 and ends at position 5000. The traverse speed is 200 RPM. ACC and DEC are specified independently before the move is commanded.

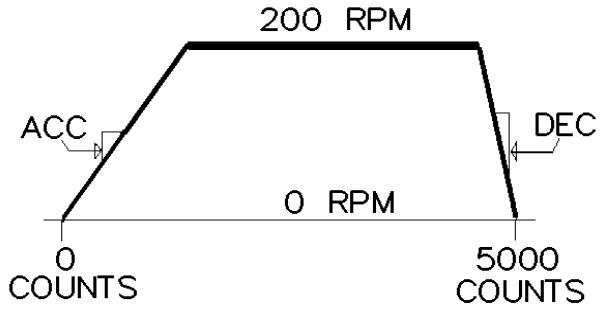


Figure 8.2 A Simple Profile

8.8.3.1 S-Curves

The VECTOR**STAR** also allows you to specify the type of acceleration you want. You can select S-curve accelerations for smoothness or straight-line accelerations for quickness. The graph in Figure 8.3 shows the profile from Figure 8.2 using S-curves instead of straight lines.

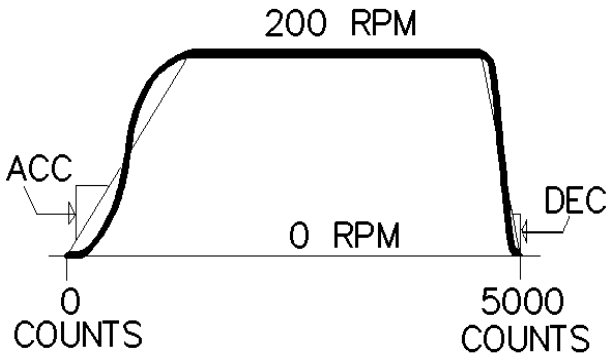


Figure 8.3 S-Curve Profile

Notice that ACC and DEC are still independent. Notice also that they specify the *average* acceleration, not the peak. Since S-curves reduce the acceleration rate at the endpoints of the acceleration, the acceleration rate in the middle must increase. Typically, when you switch to S-curves, you must reduce ACC and DEC to stay within the ratings of the motor. However, since S-curves reduce overshoot, you may find that you increase the overall acceleration rate when you use them.

You may need to reduce ACC and DEC when using S-curves.

For some applications, S-curves can reduce the average acceleration too much; in others, straight line acceleration produces motion that jerks the motor excessively. The VECTOR**STAR** provides different levels of S-curves allowing you to make the trade-off. There are five levels that are selected by setting the variable SCRV to either 1, 2, 3, 4, or 5. For more information on S-curves, see KMTG application note B101, "Acceleration Profiles."

Table 8.7 S-Curve Acceleration Chart

For this acceleration...	Set SCRV to...
Straight-Line	1
Modified Polynomial	2
Polynomial	3
Modified Sinusoid	4
Sinusoid	5

8.8.3.2 Move Absolute (MA) Command

There are two kinds of simple moves: absolute and incremental. With absolute moves, you specify the end position; with incremental moves, you specify the total distance of the move.

The MA command allows you to command absolute moves by specifying the end position. ACC, DEC, and SCRV are all in effect for MA moves. As an option, you can specify the traverse speed.

The following example moves to position 50000 at a peak speed of 1000 RPM.

```
MA 50000 1000
```

The variable VDEFAULT is the default velocity for MA and MI commands. If you enter an MA command without specifying a speed, the traverse speed will be VDEFAULT.

```
MA 100000
```

The above example would move the motor so that PFB is equal to 100000; it would assume a traverse speed of VDEFAULT. If you do not specify the speed in MA commands, it reduces the execution time. This normally means less delay between when the command is entered and when the motor begins turning. Appendix H lists the execution times of a few simple moves.

Not specifying the speed in MA commands reduces execution time.

8.8.3.3 Move Incremental (MI) Command

The MI command allows you to command incremental moves by specifying the total distance of the move. ACC, DEC, and SCRV are all in effect for MI moves. Like MA, if you enter an MI command without specifying a speed, the traverse speed will be VDEFAULT.

For example, the following command causes the motor to move 5000 counts at a peak speed of 200 RPM.

```
MI  5000  200
```

The profiles that were shown earlier as "A SIMPLE PROFILE" or "S-CURVE PROFILE" could have been generated from this example.
As with the MA command,

```
MI  25000
```

causes the motor to move 25000 counts, with the peak speed at the speed VDEFAULT. For both the MI and MA commands, not specifying speed reduces execution time and program size.

Not specifying the speed in MI commands reduces execution time.

8.8.3.4 Incremental Move Example

SHOCK HAZARD!

Large voltages from the AC line and the DC bus can cause injury. Wire the VECTORSTAR as described in Chapter 2.

THE MOTOR MAY MOVE UNEXPECTEDLY!

BE PREPARED TO DISABLE THE VECTORSTAR!

You should complet "Initial Check-Out" in the Chapter 2 before proceeding.

This section will enable the VECTORSTAR. The system may be unstable. The motor may begin oscillating or run away. Be prepared to disable the VECTORSTAR quickly by either turning off (opening the contacts of) LIMIT or REMOTE.

Turn on the AC line voltage. Type in the following example:

```
EN
ACC  1000
DEC  1000
MI  4000  100
```

This should cause the motor to rotate 4000 counts with a traverse speed of 100 RPM. With the next example the motor will repeat the move. Type:

```
VDEFAULT = 100
MI  4000
```

Notice that the motor again moves 4000 counts. To bring the motor back to the original position, type:

```
MI  -8000
```

8.8.3.5 Profile Limits

With both the MA and MI commands, if the traverse speed cannot be reached because ACC or DEC is too small for the specified move, then the VECTOR**STAR** reduces the maximum speed so that the move, for all practical purposes, is triangular. Actually, there is a very short (less than 5 milliseconds) traverse segment so that the move still has three segments.

The maximum time for an entire move is not limited. However, the time for each acceleration or deceleration is limited to 30 seconds. If the acceleration rate is so low that this limit is exceeded, then the VECTOR**STAR** generates an error explaining that either ACC or DEC is too low. This error is issued before the motion command begins. In this case ACC or DEC must be increased, or the peak speed of the move must be decreased.

8.8.3.6 Multiple Profile Commands

The VECTOR**STAR** allows one succeeding move to be calculated while the present move is being executed. This reduces inter-index delay, the delay between successive moves, almost to zero. When you are commanding motion from the Interactive mode (-->), be careful not to type in two move commands while another is executing (motion from the original command is not complete). This generates an error. If you are commanding motion from your program, the VECTOR**STAR** automatically pauses before calculating a third motion profile, thus stopping this error from occurring.

8.8.3.7 Profile Final Position, PFNL

If you want to keep track of the end position of the present move, the variable PFNL (Position Final) is provided. This variable contains the final position of a move. The variable can be used to compute the distance remaining by combining it with PFB (Position Feedback):

```
P "DISTANCE TO GO " PFNL-PFB
                ;PRINT THE AMOUNT OF
                ;POSITION TO GO TO
                ;FINISH THE MOVE
```

8.8.4 JOG (J) Command

This section describes J, the JOG command. Jogging is useful when you want to command motion without position endpoints. For example, the following command causes the motor to rotate at 500 RPM indefinitely.

```
J 500
```

Jogs are useful for machine set up and testing. ACC and DEC are in effect with Jogs, as is SCRV. Software and Hardware Travel Limits are also in effect. Jog is the only move command that can cause motion to change direction without stopping first. However, since changing direction involves both acceleration and deceleration, Jog commands that change direction of rotation use ACC or DEC, whichever is lower. Jog commands should be used with caution, since motion continues indefinitely.

8.8.5 NORMALIZE (NORM) Command

NORM, the NORMALIZE command, is required if you want to reset the VECTOR**STAR** position feedback, PFB. Often, you may want to set the position feedback to some known value. For example, on power-up the position feedback is set to zero. After a homing sequence, you may need to reset the position register. This is done using NORM. The following example sets PFB (position feedback) as well as PCMD (POSITION command) to 10000 in position units.

```
NORM 10000
```

As an alternative, you can enter:

```
PFB=10000
```

Setting PFB has the same effect as the NORM command. Use whichever you think makes your program easier to understand.

Now, type in:

```
P PFB
```

Now, normalize the position to 1000 with:

```
NORM 1000
```

Again, print PFB:

```
P PFB
```

and see that it is now 1000. The NORMALIZE command cannot be used when either GEAR is on, or when motion is commanded from MA, MI, or any other motion command.

8.8.6 Zero Position Error (ZPE) Command

The ZPE command zeros position error by setting PCMD to PFB without changing PFB. There are occasions when this will be necessary. For example, if the VECTOR**STAR** is run for some time as a velocity loop, then position error can accumulate well beyond PEMAX. If the position loop is turned on with this condition, a position error overflow error will occur. To prevent the error, you must first zero the position error, then turn the position loop on by entering:

```
ZPE
PL  ON
```

The ZPE command is also frequently used with clamping. See the explanation of clamping in Section 8.8.10

8.8.7 MACRO MOVES

This section describes functions to implement Macro moves. Macro moves are complex, user-defined moves that execute as one move. Simple moves, such as MI and MA, always begin and end at zero speed and have one acceleration segment, one deceleration segment, and one traverse segment. Macro moves allow up to 30 user definable segments for one move. The moves are fully precalculated and, therefore, can execute very fast. Like other moves, ACC, DEC, and SCRV are in effect. These parameters can be changed between Macro move segments allowing more flexibility. Also, PFNL indicates the ending position of the entire Macro move. Like MI and MA, the entire Macro move must begin and end at zero speed, although beginning and ending speeds of individual sections are not constrained to 0 RPM. Dwell segments can be embedded in Macro moves.

8.8.7.1 MCA, MCI, MCD, and MCGO

There are two kinds of Macro moves: Macro Absolute (MCA) and Macro Incremental (MCI). Dwells can be inserted using the Macro Dwell (MCD) command. When the move is completely specified, the Macro Go (MCGO) can be used to execute the move. MCGO can be executed as many times as desired, once calculations for the entire move are complete.

Both Macro Absolute and Macro Incremental moves are specified in a similar manner. You must specify either the end position (for Absolute moves) or the distance (for Incremental moves). You also can specify up to two velocities. If two velocities are specified, then the first is the traverse speed and the second is the ending speed.

If one velocity is specified, then it is assumed to be the ending speed. In this case, the VECTOR**STAR** uses the larger speed, either the beginning or ending speed, for the traverse speed. All velocities are specified greater than zero. The VECTOR**STAR** determines the direction based on the specified position. If no velocities are specified, then the VECTOR**STAR** continues the Macro section at the beginning speed until the specified position is reached.

If you want to include a dwell in the middle of a Macro move, use the Macro Dwell (MCD) command. In this command, you specify the time of the dwell in milliseconds. The following example specifies a 100 millisecond dwell.

```
MCD  100              ;100 MSEC DWELL
```

Macro dwells are only allowed at the beginning of a Macro move and when the previous section has ended at zero speed.

After all motion sections have been specified, with the final motion ending at zero speed, use the Macro Go (MCGO) command to begin the motion. MCGO is only allowed when the speed at the end of the last Macro move is 0. MCGO also ends calculations for Macro moves. Subsequent MCI, MCA, or MCD commands reset the Macro move sequence.

Subsequent executions of MCGO will execute the move again. The effect of multiple MCGO's on Incremental Macro moves is that the Incremental move is executed again. The effect of multiple MCGO's on Absolute Macro moves is more difficult to understand. This is because all Macro moves are converted to Incremental before being executed, whether they are MCI or MCA based. This can cause undesirable effects if the position does not return to the starting point at the end of the Macro move. Absolute Macro moves that are to be executed more than once should return to the starting position.

Enabling the VECTOR**STAR** resets the Macro move memory. If you are typing in a Macro move and you make an error, you should disable, then enable the VECTOR**STAR**, and retype the entire move. Jog, MA, and MI commands do not reset the Macro move memory. This means you can execute jogs or simple moves after the Macro move is calculated; the MCGO command will still execute the move properly.

8.8.7.2 Macro Move Example #1

As an example of Macro moves, consider the following profile:

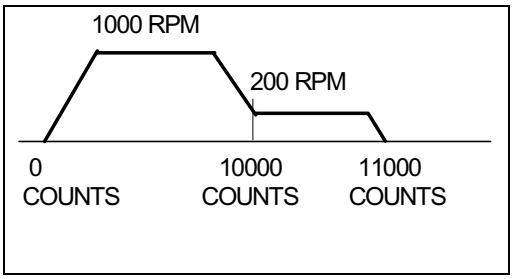


Figure 8.4 Macro Move Example #1

There is no way to use MA or MI to accomplish this profile, so Macro moves must be used. The following sequence will generate the move shown in Figure 8.4.

```
ACC=20000
DEC=20000
MCI  10000  1000  200 ;MOVE 10000
                      ;COUNTS, TRAVERSE
                      ;AT 1000 RPM AND END
                      ;AT 200 RPM.
MCI  1000   0         ;MOVE 1000 MORE
                      ;COUNTS
                      ;TRAVERSING AT
                      ;200 RPM (THE FINAL
                      ;SPEED  OF THE
                      ;PREVIOUS MOVE)
                      ;AND END AT 0 RPM.
MCGO                  ;BEGIN MOTION
```

Every subsequent MCGO will generate a similar move, 11000 counts long.

8.8.7.3 Macro Move Example #2

The profile can be made slightly more complex by adding a 0.5 second dwell and a return to the original position on the end. This profile is demonstrated in Figure 8.5. Note that this diagram is a shorthand "schematic" of motion. This curve is plotted as velocity-versus-time for forward motion (the first 5 segments) and for the dwell. However, return motion is shown as negative motion returning to the origination time. Obviously, time does not go backwards. This method of diagramming motion is commonly used because it is simple (if not in all respects accurate) and conveys the necessary information.

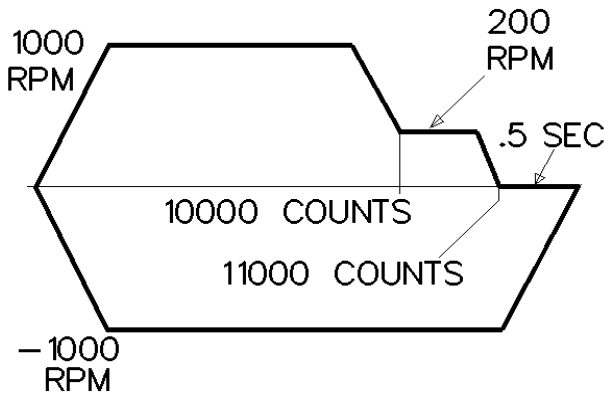


Figure 8.5 Macro Move Example #2

The above sequence should be modified as follows. Note that here the profile is converted to Absolute rather than Incremental—this is a matter of convenience as either will work.

```
MCA 10000 1000 200
MCA 11000  0
MCD 500               ;ADD A DWELL
MCA 0 1000 0          ;RETURN TO HOME.
                      ;NOTE THAT VELOCITY
                      ;IS ALWAYS POSITIVE
MCGO
```

Note that Macro moves have one inherent weakness. If you are using user units and you specify an incremental move that translates to a non-integer number of counts, the Macro move will move the closest number of integer counts. If the move is repeated, the small error in the position command will accumulate. This problem does not happen if you use MI commands.

8.8.8 R/D BASED MOVE (MRD) Command

This section describes MRD, the command that generates moves based on the feedback from the R/D converter, rather than the Position command (PCMD). These moves are less than one revolution and are always Absolute, rather than Incremental.

With the MRD command, you specify the desired R/D output (at the end of the move) and the peak velocity. For example, the following command moves the motor so that the R/D output is 1000.

```
MRD  1000  100
```

100 RPM is the traverse speed. ACC, DEC, and SCRV are all in effect with MRD. As with MI and MA, if 100 RPM is too large to be attained given ACC and DEC, the move becomes triangular.

As an option, directions of CW or CCW can be specified to force the motor to rotate the desired direction. If direction is left out, then the motor rotates whichever direction is shortest.

```
MRD  1000  100  CW  ;MOVE R/D TO 1000,
                    ;BUT ALWAYS CW
```

The above example moves the motor clockwise, even if the specified position (1000) is just a few counts counter-clockwise. The variable DIR has no effect on MRD commands.

The limit of position is based on the R/D converter accuracy as shown in Table 8.8.

Table 8.8 R/D Converter Accuracy

Resolution	Maximum Position
12	4095
14	16383
16	65535

MRD moves are not buffered. They are not allowed when the VECTOR**STAR** is jogging or if a move is in progress.

MRD moves can be used to improve the accuracy of homing sequences. First, use the VECTOR**STAR** to position the motor as close as possible to the home limit switch trip point. Then, use the MRD command to move the motor to a specified R/D position. In this case, the limit switch must be accurate only to one-half revolution of the motor for the R/D moves to be useful.

8.8.9 Capturing Position

Position capture is a feature where the position feedback (PFB) is *captured* when a hardware input transitions. The VECTOR**STAR** position capture is accurate to +/-25 microseconds. In other words, the position that is stored after a capture is equal to the actual position of the motor at the time of the capture, within 25 microseconds. Capture uses the HOME hardware input as the capture trigger.

8.8.9.1 Enabling Capture: CAP and PCAP

The switch CAP controls capture. If CAP is on, then capturing is enabled. When capturing is enabled, the VECTOR**STAR** will watch the HOME input. When the HOME input changes to the state specified by CAPDIR, the VECTOR**STAR** will store the position in the variable PCAP. After the capture, the VECTOR**STAR** turns CAP off. This tells you that the capture is complete. PCAP is in position units. You can then use PCAP as you would any other monitoring variable.

8.8.9.2 Capture Direction, CAPDIR

The capture is triggered when the HOME input changes from 0 to 1, or vice versa. If CAPDIR is 1, the capture occurs when the HOME input changes from 0 to 1. If CAPDIR is 0, the capture occurs when HOME changes from 1 to 0. CAPDIR can be changed at any time. Changing CAPDIR always turns CAP off.

8.8.9.3 Speeding Up Homing Sequences

One application of capture is to speed up homing sequences. Homing sequences traverse very rapidly until the HOME switch is tripped. Then the motor decelerates to zero and begins to traverse at a medium speed in the opposite direction until the HOME switch trips again. Then the motor decelerates again to a slow speed until the HOME switch trips again. Since the final speed was low, the distance to decelerate is considered negligible, and the motor is assumed to be at home.

Using capture, the approximate home location can be determined when the motor is traversing at high speed, eliminating the need for the medium speed traverse. The following program illustrates this.

```
CAPDIR=1
CAP  ON
J  -5000              ;JOG AT -5000 RPM
                      ;TO GET TO HOME
TIL  CAP  EQ  0       ;WAIT FOR CAPTURE
                      ;TO OCCUR
J  0                  ;STOP MOTION
MA  PCAP  200         ;RETURN TO PCAP—
                      ;APPROXIMATE HOME
J  1                  ;JOG AT A LOW
                      ;SPEED TIL HOME
                      ;CAN BE FOUND
TIL  HOME  EQ  0      ;ONCE HOME IS
                      ;CROSSED, STOP.
J  0
```

The capture position is accurate to 25 microseconds. The resulting error is proportional to speed. For example, for a 12-bit R/D converter, if the capture were done while the motor was rotating at 5000 RPM, the error would be limited to about 1 degree. If this is not close enough, you can jog the few bits until the switch is tripped, or you can use the MRD as discussed above.

8.8.10 Clamping

Clamping stops VECTOR**STAR** motion when the position error exceeds a set point. This is used to determine that the motor, usually through a lead screw, has run a part into a mechanical stop. The profile stops and the part is held with limited torque. This is sometimes referred to as "Feed to Positive Stop." The stop is detected by watching position error; when position error exceeds the variable PECLAMP, the part is assumed to have run into a stop. When a stop has been detected, the VECTOR**STAR** will hold the current at ILIM which should be set to the proper holding current. ILIM can be increased or decreased after the stop has been detected. To enable clamping, turn CLAMP on. PECLAMP can be changed at any time.

In general, clamping is done at low speeds with the current limited to some low level. After the clamp has occurred, the motor is assumed to be at zero speed. When the clamp has occurred, you can raise or lower ILIM to set the holding torque as desired. You can tell whether a clamp has occurred by looking at SEG, the present motion segment. If SEG is 0, then motion has stopped.

After the VECTOR**STAR** stops motion, the position error stays at approximately PECLAMP. Before commanding any new motion, you should zero the position error with the ZPE command.
Clamping can be used with all move and jog commands. If jogs are used, the motion continues until the stop is found. If move commands are used, then motion does not continue past the specified endpoint, regardless of whether a part is found.

An example of clamping follows:

```
PECLAMP=1000        ;SET CLAMP  = 1000
                    ;POS UNITS
CLAMP  ON           ;ENABLE CLAMPING
                    ;MODE
MA  100000  400     ;MOVE AT MOST
                    ;100000 POS UNITS
                    ;IF THE MOTOR
                    ;GETS ALL THE
                    ;WAY TO 100000,
                    ;THEN THE STOP
                    ;WAS NOT
                    ;ENCOUNTERED.
                    ;ASSUMED
                    ;THE PART IS NOT
                    ;THERE.

W 0                 ;DELAY UNTIL
                    ;MOTION STOPS

IF  PCMD  EQ  100000  P  "PART NOT
FOUND"
                    ;IF PCMD = 100000 =
                    ;FINAL POSITION,
                    ;THEN THE PART
                    ;WAS NOT FOUND.
```

8.8.10.1 Clamping and Homing

Clamping can be used to home your machine by gently running the machine into a stop; this eliminates the need for a home limit switch. In this case, you should reduce ILIM to a level just high enough to overcome running friction at low speed. ILIM is lowered to reduce the torque exerted by the motor when the machine stop is encountered. Set PECLAMP to a level well above the normal following error; usually the position unit equivalent of several hundred counts is sufficient. Then turn CLAMP on and jog, at low speed, toward the stop. The VECTOR**STAR** will run the machine into a stop and limit current to ILIM. When SEG is equal to 0, the VECTOR**STAR** has clamped and thus recognizes that the machine has been run into the stop.

Often, the repeatability of this operation is unacceptable because the stop may be "soft" or it may wear over time. Here, you can use the MRD command to force the VECTOR**STAR** to move to a fixed R/D converter position. This means that you will get a repeatable home as long as the clamp position does not vary more than one-half of one revolution between different clamping operations. This is not normally a problem.

To set the proper R/D converter position for the MRD command, first do the clamping operation by hand a few times. Reduce ILIM and jog, at low speed, into the stop. After the unit has clamped, as indicated by SEG = 0, print the R/D converter position using:

```
P  PRD
```

Do this several times and record the average of PRD. Now use the MRD command to back away from the stop about one-half of one revolution. For example, suppose you jog clockwise into the stop several times and record

PRD each time. It turns out that the average value of PRD is 1500 counts. Then use the following MRD command:

```
MRD 1500+2048 200 CCW  ;MOVE TO 1/2
                       ;REVOLUTION
                       ;FROM 1500
                       ;COUNTS
```

You must specify the direction (CW or CCW) so that the VECTOR**STAR** always backs away from the stop. Remember that, for example, *J 1000* is not necessarily clockwise since the direction of jog rotation is controlled by the variable DIR.

You should be aware that if you replace your motor, you must repeat this process since the relationship of PRD to the motor shaft position is different for each motor.

NOTE

If you replace your motor, repeat this process.

8.8.11 JOG TO (JT) & JOG FROM (JF)

In some applications, JOG commands need to be synchronized with position feedback. With J, the standard JOG command, the speed changes when the command is entered. Position dependent jogs (Jog To and Jog From) delay the speed change until a specified position is reached. You specify the position at which the change in speed begins with the Jog From (JF) command. Similarly, you specify the position at which the change in speed ends with the Jog To (JT) command. With position dependent jogs, you must specify a position and the new speed. ACC, DEC, and SCRV are in effect. Position dependent jogs are always Absolute moves (not incremental).

The following graph shows the effect of a JF command. This example assumes that the speed is already 2000 RPM when the JF command is executed.

```
;ASSUME PRESENT SPEED IS 2000 RPM
JF  50000  1000
```

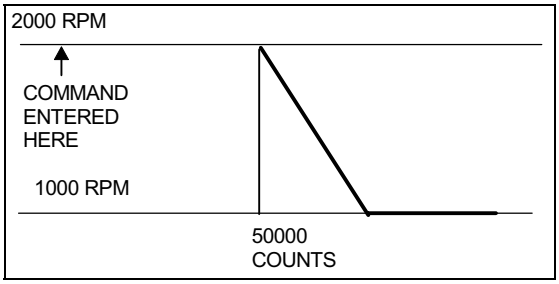


Figure 8.6 Jog From (JF) Command

The next graph shows the effect of the Jog To (JT) command. This example also assumes that the speed is 2000 RPM when the command is executed:

```
;ASSUME PRESENT SPEED IS 2000 RPM
JT  50000  1000
```

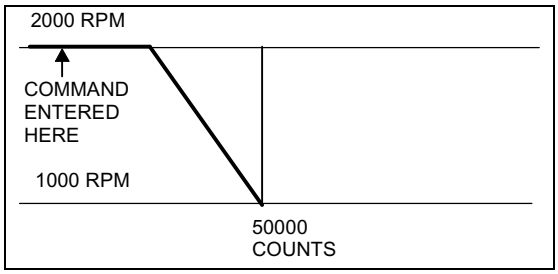


Figure 8.7 Jog To (JT) Command

Position dependent commands must be used with care. If you specify a position that has already passed, the VECTOR**STAR** will generate ERROR 42, "MOVE W/ O TIME." Also, if the Jog To command is given so that ACC or DEC prohibits the profile from reaching final speed before the specified position, the VECTOR**STAR** will generate ERROR 42. ERROR 41, "MOVE NEEDS MOTION," is generated if Jog To or Jog From are commanded when the velocity is 0. Finally, a position dependent jog that attempts to change the direction of rotation will generate an error. All of these errors stop motion.

8.8.11.1 Registration

The VECTOR**STAR** allows you to combine the position capture with the Jog To command to implement index-to-registration. One example of index-to-registration is a conveyor belt on which items are placed in random positions. An optical sensor detects the item upstream of the operation. The VECTOR**STAR**, controlling the conveyor, continues at full speed and stops the item where the operation will take place. The high-speed position capture works at all velocities and during accelerations. It is accurate to 25 microseconds (if Connector C2, Pin 19 is used) and, therefore, will work properly on demanding index-to-registration applications. If the OPTO-22 Connector (C7) is used with standard industrial OPTO-22 style modules, the optical module may add as much as 25 milliseconds of delay, so be careful to properly specify the optical coupling to the registration switch.

To implement index-to-registration, you usually jog the motor at a constant speed, capture the position (with the registration device connected to the HOME input), then use the Jog To command to stop the motor at an endpoint (normally a specified distance beyond the registration input).

8.8.11.2 Registration Example

The following example shows how to program the VECTOR**STAR** for registration. The desired operation of the program is as follows:

1. Set CAPDIR (1 for low-to-high transition, 0 for high-to-low transition).
2. Enable capturing.
3. Begin move.
4. Wait for the VECTOR**STAR** to capture.
5. Use the captured position to set the endpoint of the move.

For example, the following code segment jogs at 2000 RPM and stops 4000 counts after the registration input transitions from low to high.

```
CAPDIR  1              ;SET CAPDIR FOR
                       ;LOW TO HIGH
CAP  ON                ;ENABLE CAPTURE
J  2000                ;BEGIN MOVE
TIL  CAP  EQ  0        ;WAIT FOR POSITION
                       ;CAPTURE
JT  PCAP+4000  0
```

Note that the motor comes to rest 4000 counts after the position that was captured, not 4000 counts after the JT command is executed. If 4000 counts was not enough distance, ERROR 42, "MOVE W/O TIME," would be generated. This means that the commanded speed change cannot be accomplished given DEC, the deceleration limit. Note also that you must leave an additional 10-15 milliseconds for the TIL and JT commands to be executed.

The JT command example given here brings the system to rest. As an alternative, you can change the speed to any value the motor can run, as long as you do not attempt to change direction with one JT command. For example, the following command replaces the above JT command when you want to change speed to 100 RPM at 4000 counts past PCAP.

```
JT  PCAP+4000  100
                       ;CHANGE SPEED TO
                       ;100 RPM.  BEGIN
                       ;DECEL SO THE SPEED
                       ;IS JUST REACHING
                       ;100 RPM WHEN THE
                       ;POSITION IS 4000
                       ;COUNTS PAST
                       ;REGISTRATION MARK
```

For more information about registration, see KMTG application note "Cut to Length."

8.8.11.3 Multiple JF/JT Commands

Many applications require that multiple Jog From (JF) and Jog To (JT) commands be executed sequentially. In most cases, you will have to insert a delay in your program between JT and JF commands. For example, if you enter:

```
55$
EN                     ;ENABLE
                       ;VECTORSTAR
ACC  100000            ;SET ACCEL AND
                       ;DECEL RATES
DEC  100000
NORM  0                ;NORMALIZE TO
                       ;ZERO POSITION
J  100                 ;JOG TO 100 RPM
JT  20000  400         ;ERROR—SHOULD
                       ;DELAY TIL SPEED
                       ;REACHES 100 RPM
                       ;BEFORE
                       ;EXECUTING JT
                       ;COMMAND.
```

```
JT  30000  0          ;ERROR—SHOULD
                      ;DELAY TIL SPEED
                      ;REACHES 400 RPM
                      ;BEFORE EXECUTING
JF                    ;COMMAND.
DIS
B
```

You might think the motor will first jog to 100 RPM, then to 400 RPM (at 20000 counts) and finally come to rest at 30000 counts. Actually, the motor will jog to about 40 RPM and continue at that speed until it comes to rest at 30000 counts. This is because the JF/JT commands cause the motion profile to hold the velocity command constant, even if an acceleration is commanded from the previous motion command. The solution is to insert delays to force the program to wait until the motor reaches the final speed from the previous motion command. For example, the above program can be modified as follows:

```
55$
EN                    ;ENABLE
                      ;VECTORSTAR
ACC  100000           ;SET ACCEL AND
                      ;DECEL RATES
DEC  100000
NORM  0               ;NORMALIZE TO
                      ;ZERO POSITION
J  100                ;JOG TO 100 RPM
TIL  VCMD  EQ  100    ;WAIT TIL SPEED
                      ;REACHES 100 RPM
JT  20000  400        ;EXECUTE JT
                      ;COMMAND
TIL  VCMD  EQ  400    ;WAIT TIL SPEED
                      ;REACHES 400 RPM
JT  30000  0          ;EXECUTE JT
                      ;COMMAND
DIS
B
```

Although delays with the TIL command work, delays usually should be inserted with the WAIT (W) command. The WAIT (W) command takes less space and works better with multi-tasking, a subject discussed in Chapter 9. For our example, the first TIL command can be replaced with “W 2” and the second can be replaced with “W 3.”

8.9.11.4 Changing Profiles During Motion

Position dependent jogs can also be used to change the speed or endpoints of an MA, MI, MCI, or MCA command that is already in progress. For example, suppose you want to change the speed of a profile depending on an input. You could write the following program to reduce the speed when I1 is 1.

```
X1 = 10000            ;X1 STORES THE
                      ;ENDPOINT
MA  X1  5000          ;BEGIN AT 5000 RPM
TIL SEG EQ 0 GOSUB 25
                      ;25$ WATCHES I1
                      ;TO CHANGE SPEED
B

25$
? I1  EQ  0  RET      ;CHANGE ONLY IF
                      ;I1 = 1
J  1000               ;REDUCE SPEED
                      ;TO 1000 RPM
TIL  SEG  EQ  2       ;WAIT UNTIL SPEED
                      ;IS 1000 RPM
JT  X1  0             ;USE JT TO GET TO
                      ;ORIGINAL ENDPOINT
                      ;AT NEW SPEED
TIL  SEG  EQ  0       ;WAIT FOR MOTION
                      ;TO STOP
B                     ;DONE
```

You must be careful not to begin the motion too late in the profile. For example, suppose I1 became 1 after the profile was well into deceleration, and the speed was, say 200 RPM. In this case, the JT command would generate an error because by the time it was executed, the motor position would be past X1, the original endpoint. This is because the unit would accelerate up to 1000 RPM before the JT command was executed. In general, you must limit the time during which you are looking for the speed change. After this point, the profile must either continue along the original profile or the endpoint must be extended. For example, the program section beginning at label 25 could be re-written so that it watched a position trip point, X1-2000:

```
25$
? PFB  GT  X1-2000  RET    ;DO NOT
                           ;REDUCE
                           ;SPEED IF
                           ; PFB >
                           ;SETPOINT
;
;REST OF 25$ PROGRAM THE SAME
;
```

What value to use for the setpoint varies from one application to another. These values must be set by experience. On many applications, the input will not request a speed reduction near an endpoint, so that this may not be a problem.

8.8.12 External Inputs

External inputs are normally from a "master" motor or analog input. As a standard, these inputs are in digital encoder format. Examples of "master" motors include the encoder-like output from another VECTOR**STAR**, output from an actual encoder, or a customer synthesized encoder signal. The external input can control motion in the two VECTOR**STAR** Master/Slave modes: electronic gearbox and profile regulation. The VECTOR**STAR**, acting as the slave, accepts commands from these external sources. The external input can also come from a feedback encoder which is mounted to the motor; this encoder is occasionally used to improve the accuracy of the VECTOR**STAR**.

External inputs are connected to Channel A and B inputs of the Encoder Equivalent Connector.

Your program has direct access to the external input through the variables VEXT and PEXT. The frequency of the external input is provided in VEXT. VEXT is in external velocity units (VXNUM and VXDEN). PEXT is the accumulation of counts from the external input. PEXT can be set to any value from the terminal or from your program at any time; this is equivalent to normalizing the external position. PEXT is in external position units (PXNUM and PXDEN). If the external input comes from a motor, VEXT and PEXT represent the "master" motor's velocity and position, although you must properly calculate the external velocity and position units. In this way, PEXT, the master position, is similar to PFB, the slave position. Likewise, VEXT is similar to VFB. If the "master" motor has the same resolution as the slave, then set PXNUM, PXDEN, VXNUM, and VXDEN equal to PNUM, PDEN, VNUM, and VDEN, respectively. Otherwise, see Chapter 9 for more information on calculating the units.

VXAVG is the average of VEXT over the previous 16 milliseconds. Occasionally, the normal sample-to-sample variation of VEXT is undesirable. In these cases, use VXAVG in place of VEXT.

8.8.12.1 Analog Input

Also, the VECTOR**STAR** provides an analog external input. Note, however, that you cannot have both types of inputs at the same time. The VECTOR**STAR** features an on-board A/D 12-bit converter. The analog input is on connector C10, pin 5 (HI) and pin 6 (LO). (Connector C2 can also be used as analog input if the option card is installed.) A switch called A2D is used to switch the gear input. If A2D = 1, the signal is from the analog input; if A2D = 0, the signal comes from the encoder input.

If the analog input is a velocity command, then use electronic gearbox Master/Slave mode to make the VECTOR**STAR** a velocity drive. If the analog input is going to be used for "feedrate override," use profile regulation.

The analog external input is connected to the customer I/O.

8.8.13 Electronic Gearbox

Electronic gearbox is one of two VECTOR**STAR** Master/Slave modes. Refer to Figure 8.8 for a diagram of the two modes. Electronic gearbox is used to link two motors together so that the velocity of one is proportional to the velocity of the other. The constant of proportionality can be negative, allowing the velocities to be in opposite directions.

If the analog input is used, a gear ratio calculation is not necessary. Simply give a value to VSCALE, which is the velocity input scale factor. The value entered is the desired motor velocity of 10 volts applied to the analog input.

8.8.13.1 Gear Ratio, GEARI & GEARO

If input is in digital encoder format, you must calculate the gear ratio.

In electronic gearbox, the command signal comes from the external input. The pulses are multiplied by a gear ratio to form the position or velocity command. The ratio is defined by two variables: input gear teeth (GEARI) and output gear teeth (GEARO). GEARI must be between ±32767; GEARO must be between 1 and 32767. If the sign of GEARI is changed, then the direction of rotation will be reversed.

If the master is a motor or encoder, calculate GEARI and GEARO with:

$$\frac{\text{GEARI}}{\text{GEARO}} = \frac{\text{REV}_{\text{SLAVE}}}{\text{REV}_{\text{MASTER}}} \times \frac{\text{RESOLUTION}_{\text{SLAVE}}}{\text{RESOLUTION}_{\text{MASTER}}}$$

where:

REV_{MASTER} is an arbitrary number of revolutions of the master motor,

REV_{SLAVE} is the corresponding number of revolutions of the slave motor,

$RESOLUTION_{SLAVE}$ is the resolution of the slave motor in counts/revolution, and
$RESOLUTION_{MASTER}$ is the resolution of the master motor in counts/revolution.

If the master is a pulse train that does correspond to a motor or encoder, calculate GEARI and GEARO with: where:

$COUNTS_{MASTER}$ is an arbitrary number of counts of the master signal and

REV_{SLAVE} and $RESOLUTION_{SLAVE}$ are as before.

If the signal is from the analog input (A2D = 1), the gear ratio will be

GEARI/GEARO = 700/16384 for
10 volts = 10000 rpm velocity command.

Clearly,

GEARI/GEARO = 70/16384 will give you
10 volts input = 1000 rpm.

To enable the Gearbox mode, type:

GEAR ON

If the ratio is not an integer, the VECTOR**STAR** does not "drop pulses." The VECTOR**STAR** keeps track of partial pulses to eliminate dropping pulses over time. If the number of pulses coming into the VECTOR**STAR** is at a rate that is too large, then ERROR 97, "GEAR OVERFLOW," will be generated. This error can also be caused by the ratio of GEARO to GEARI being too large. Note that large feed-forward (KF > 4000) is normally undesirable in electronic gearbox systems because it causes overshoot.

8.8.13.2 Gearbox Example 1

Two VECTOR**STAR**s are connected in a master/slave system. Both have 12-bit R/D converters so that one revolution is equivalent to 4096 counts. Suppose we want the slave motor to rotate at one third the speed of the master motor. What are the values of GEARI and GEARO?

$$\frac{GEARI}{GEARO} = \frac{REV_{SLAVE}}{REV_{MASTER}} \times \frac{RESOLUTION_{SLAVE}}{RESOLUTION_{MASTER}}$$

$$\frac{GEARI}{GEARO} = \left(\frac{1}{3}\right) \times \left(\frac{1}{1}\right) = \frac{1}{3}$$

You can select any integer values for GEARI and GEARO that have the ratio 1/3.

8.8.13.3 Gearbox Example 2

Suppose the master signal in Example 1 came from a 500-line encoder. With quadrature encoding, a 500-line encoder will generate 2000 counts per revolution. If you

$$\frac{GEARI}{GEARO} = \frac{REV_{SLAVE} \times RESOLUTION_{SLAVE}}{COUNTS_{MASTER}}$$

still wanted 1:3 gearing, then:

$$\frac{GEARI}{GEARO} = \frac{REV_{SLAVE}}{REV_{MASTER}} \times \frac{RESOLUTION_{SLAVE}}{RESOLUTION_{MASTER}}$$

$$\frac{GEARI}{GEARO} = \left(\frac{1}{3}\right) \times \left(\frac{4096}{2000}\right) = \frac{4096}{6000}$$

So, GEARI would be 4096 and GEARO would be 6000.

8.8.13.4 Profiles and Gearbox

Gearboxing can be done in conjunction with incremental moves and jogs. MI and Macro moves based on MCI are summed with the gearbox command to form the profile. This can be used for "phase adjustment," a common function used with electronic gearbox. Phase adjustment means that the slave will be locked to the master through the electronic gearbox, but occasionally the slave VECTOR**STAR** adds a short profile on top of the gearbox command. For example, you may want to increase the slave position (phase) by 90° while remaining in gear. In this case, enter the following commands:

```
                ;GEARBOX
;
; ...NORMALLY, SOME TIME WOULD ;PASS
BETWEEN THESE COMMANDS...
;
MI 1028 10    ;PHASE ADJUST 90
              ;DEGREES AT 10 RPM.
              ;SYSTEM REMAINS IN
              ;GEARBOX THROUGH THE
              ;PHASE ADJUSTMENT.
```

You cannot use MA or MCA commands when GEAR is on. Also, you cannot use position-dependent jogs (JT or JF) when GEAR is on.

8.8.13.5 Velocity Offset, VOFF

VOFF, velocity offset, is added to the Velocity command when the gearbox is enabled. VOFF is in velocity units. It is normally used with the analog input to correct voltage offset in the optional analog velocity input. VOFF can be changed at any time. Note that VOFF is set to zero when GEAR is enabled. This is done because if VOFF is large (say, 2000 RPM), enabling the gearbox would immediately command motion.


NOTE

VOFF is set to zero when GEAR is turned on.

8.8.13.6 Gearbox, ACC/DEC, and Jogs

When the VECTOR**STAR** is run as a velocity loop (PL off), acceleration and deceleration rates can be limited by the variables ACC and DEC. This allows you to limit the acceleration from external velocity commands that are otherwise unlimited. If you want the acceleration and deceleration to be limited by ACC and DEC, type:

```
RAMP ON       ;LIMIT ACC AND
              ;DEC WHEN PL IS OFF
```

8.8.14 Profile Regulation

This section describes profile regulation, one of the VECTOR**STAR** Master/Slave modes. Profile regulation allows you to synchronize the rate of profile execution according to the external input. This modifies the velocity and acceleration of move commands without affecting the final position of the move. The rate of the move is dependent on the frequency of an external clock, which is connected to the external input, in addition to the normal limits of the move (ACC, DEC, and the velocity are set in the move command itself). The external input may be a master motor to which all moves must be synchronized (such as a conveyor belt motor), or it may be a signal that you generate electronically. As an option, an analog signal can be fed directly to the VECTOR**STAR**, where it is converted to a pulse train and can be used as the external input. Profile regulation works with MA and MI, as well as Macro moves.

All profile regulation is based on an accumulation of counts from the external input during the move. If the external frequency changes during a move, the velocity of that move will be proportional to the changing clock frequency. In fact, if the external input frequency goes to zero, then motion will stop. Note that if the external input changes rapidly, the profile is not limited to ACC or DEC. For example, if the external frequency stopped suddenly, the VECTOR**STAR** would command motion to stop just as suddenly. Note also that large feed-forward (KF > 4000) is normally undesirable during regulation because it causes overshoot.

8.8.14.1 REG and REGKHZ

REG enables the Profile Regulate mode. If REG is on, then profile regulation is enabled. REG and GEAR cannot be on at the same time.

To use profile regulation, you must determine:

1. The maximum frequency of the external input. Set REGKHZ to this value.

2. The desired speed of the move when the external input frequency is REGKHZ. Use this value as the commanded velocity of the profile.

The maximum frequency of the external input is stored in the variable REGKHZ in kHz. The profile will execute normally (that is, at the specified velocity and acceleration) when the external input frequency is equal to REGKHZ. If the input frequency is less than REGKHZ, then the profile will move the specified distance, but the acceleration and velocity will be less than, and in proportion to, the input frequency. The move will never go faster than specified in the original move command, even if the input frequency goes above REGKHZ. However, the input frequency should always be less than REGKHZ. REGKHZ is only resolved to 1 kHz (for example, 499.5 kHz is converted to 500 kHz).

REGKHZ is somewhat arbitrary; it must be greater than the maximum frequency of the external input and less than 2 MHz. Beyond those limits you can set it to any frequency that is convenient and adjust the commanded motion by changing the speed of the profile.

The frequency of the external input should always be less than REGKHZ.

8.8.14.2 Profile Regulation and Counting Backwards

In general, if you use profile regulation, the external input should count forward (that is, VEXT should be positive when VXNUM and VXDEN are positive). The profile regulation firmware allows the input to count backwards for up to 30000 counts. This is useful for applications such as conveyor belts that generally go forward but can go backward for short distances. If the external input counts backwards, the Profile Regulation mode works as follows:

- The profile stops (no motion is commanded) during backward counting.
- The backward counting must be limited to 30000 counts. Otherwise, ERROR 64 is generated.
- The profile does not continue as soon as forward counting begins. The forward counts must completely offset the backward counts before the profile will continue.
- At the point where forward counts offset backward counts, the profile continues as if the input had never gone backwards.

Profile Regulation works with standard moves (MA, MI, and MRD), Macro moves, and all jogs (J, JT, and JF).

8.8.14.3 Regulation Example

A machine has an axis that operates on parts passing by on a conveyor belt. The profiles executed by the motor must be at a rate proportional to the conveyor belt speed. The belt moves at about 200 inches/minute. An encoder has been placed on the conveyor, and the maximum belt speed of 275 inches/minute is equivalent to 780 kHz on the encoder. If the belt is at maximum speed, the profile of the motor is to rotate one revolution at a peak speed of 400 RPM.

Solution: Connect the conveyor belt motor encoder to the input channel of the VECTOR**STAR**, as shown in the Chapter 2, "Wiring C1." The following program should be executed:

```
REG  ON              ;ENABLE PROFILE
                     ;REGULATION
REGKHZ=780           ;SET THE MAX
                     ;EXTERNAL
                     ;FREQUENCY TO 780
MI  4096  400        ;MOVE ONE
                     ;REVOLUTION AT
                     ;400 RPM
```

In the case above, the MI move will generate a one-revolution move at a speed proportional to the external input frequency with 400 RPM the maximum rate when the external input frequency is 780 kHz.

Note that the belt speed virtually never reaches 275 inches/minute. However, REGKHZ must be higher than the worst case maximum belt speed. For example, the above program can be modified to allow an even larger belt speed.

```
REG  ON              ;ENABLE PROFILE
                     ;REGULATION
REGKHZ=1560          ;SET THE MAX
                     ;EXTERNAL
                     ;FREQUENCY
                     ;TO 1.56 MHZ
MI  4096  800        ;MOVE ONE
                     ;REVOLUTION AT
                     ;800 RPM
```

Notice that REGKHZ was doubled. However, since the speed of the move was also doubled to 800 RPM, the commanded move is identical.

8.8.15 Encoder Feedback

Some special applications demand more accuracy than can be provided with a resolver based system. For these cases, you can mount an encoder to the motor and feed the encoder's output into the external input. The requirements for such a system are:

1. The resolution of the encoder must match the resolution of the resolver on your VECTOR**STAR** system. Refer to the Chapter 1 and the model number to determine the resolution of your system. Select the encoder as follows:

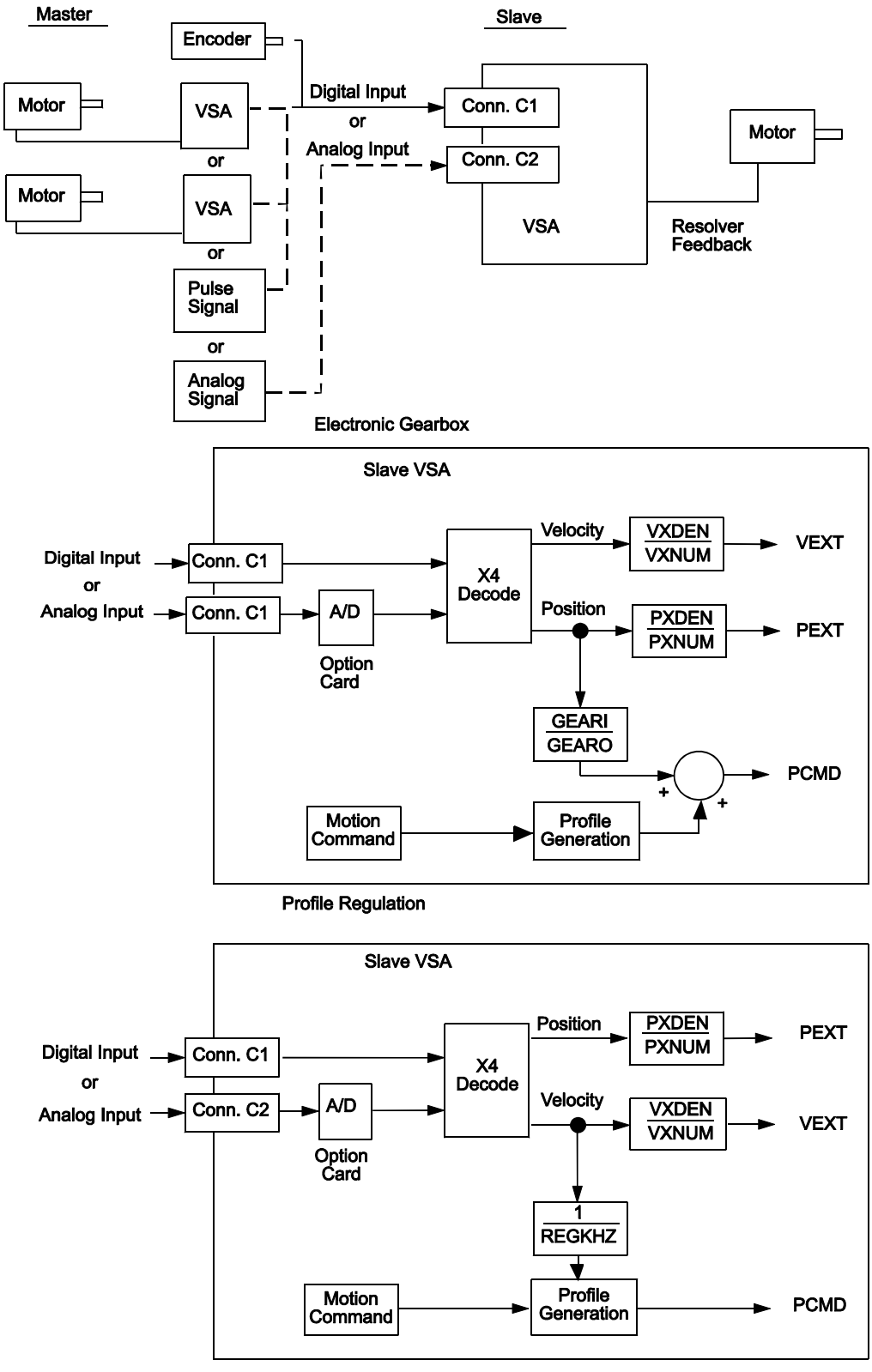


Figure 8.8 VECTORSTAR Master/Slaving

Table 8.9 Encoder Resolution

R/D Resolution	Encoder Lines/Revolution
12-bit	1024
14-bit	4096
16-bit	16384

2. The encoder must be mounted directly to the motor. It cannot be connected through gearboxes, lead screws, or any other mechanical device.

3. You must turn the switch EXTLOOP on. This switch configures the VECTOR**STAR** to close the position loop with feedback from the external input rather than from the resolver.

When EXTLOOP is on, PE, the position error, is the difference of PCMD and PEXT, rather than the difference of PCMD and PFB. The ZPE command zero's the difference of PCMD and PEXT. Also, the NORM command normalizes both PEXT and PFB simultaneously.

8.8.16 CONTINUE

The CONTINUE command is provided as a controlled way to turn off master/slave position control. The CONTINUE command tells the VECTOR**STAR** to keep the motor going at its present speed while simultaneously turning off REG and GEAR. One use of this command is to cause a controlled deceleration to 200 RPM, for example, when the electronic gearbox is enabled. If you just typed:

```
J 200
```

it would have the effect of adding 200 RPM to the command from the gearbox. However, if you typed:

```
CONTINUE
J 200
```

CONTINUE would disable the electronic gearbox while commanding the motor to continue at whatever speed it was going when the command was executed. Then the *J 200* command would bring about a controlled deceleration to 200 RPM.

CONTINUE normally looks at the velocity command for 1 millisecond. If the velocity command is generated from the electronic gearbox or a regulated profile, the velocity can vary considerably. The CONTINUE command allows you to specify a time period, up to 1 second, over which velocity command is averaged. For example, if you entered:

```
CONTINUE 50
```

CONTINUE would change the velocity command to the average velocity command over the previous 50 milliseconds. CONTINUE always sets SEG to 1.

The VECTOR**STAR** provides several control loops. These loops, or control algorithms, allow you to select the best control method for your applications.

8.9 CONTROL LOOPS

There are four sections of control loops that are of interest: input, output, feedback, and tuning variables. The input is compared to the feedback to generate an error. The error signal is modified using the tuning variables to generate the output. The tuning variables can be modified to produce higher levels of performance; unfortunately, higher performance brings with it greater noise susceptibility and reduced stability. The system designer must optimize noise and performance for the application.

VECTOR**STAR** control loops have one or two tuning variables. All VECTOR**STAR** loops follow the convention that larger constants provide higher gain. Each loop is described below and shown in Figure 8.9 at the end of this chapter.

8.9.1 Position Loop

The Position Loop input is the variable PCMD, the position command. The feedback is PFB, the position feedback. The output is VCMD, velocity command, and its two tuning variables are KP, the position loop gain, and KF, the position loop feed-forward gain.

The position loop calculates the position error (PE) as the difference of PCMD and PFB. As a secondary command source, PCMD is differentiated (d/dt)PCMD. The position loop then performs the following calculations:

VCMD=KP_PE+KF_(d/dt)PCMD

The position loop is optional. If the switch PL is on, then the position loop is enabled; if it is off, then the position loop is bypassed. PL is turned on at power-up.

The feed-forward gain reduces position error at high speed. Without feed-forward, the velocity command is generated only from position error; a large position error is required to command a high speed. If KF is large enough, then a high velocity command can be generated with little or no position error. The VECTOR**STAR** scales KF so that unity feed-forward occurs when KF equals 16384. In other words, if KF is 16384, no position error is required to generate the velocity command in steady-state running conditions. KF should never be larger than 16384. In addition, larger KF makes the system more responsive to commands.

Unfortunately, large values of KF cause overshoot. KP must be reduced to reduce overshoot. If you need to minimize position error when the motor is turning, you will need to optimize KF and KP. Typically, KF ranges from 2000 to 10000.

TQ should be off when PL is turned on. The system becomes unstable when PL and TQ are both on. If you do not turn TQ off before turning PL on, the VECTOR**STAR** will force TQ off.

When PL is turned on, TQ is turned off automatically.

8.9.2 Velocity Loop

The velocity loop takes its input from the position loop if PL is on. If PL is off, motion commands directly control the velocity command (VCMD). The feedback is VFB, velocity feedback, and the difference of these two signals is VE, velocity error. Velocity error can be used in two control loops: proportional and integrating.

8.9.2.1 Proportional Velocity Loop

If a proportional velocity loop is selected, then the velocity error is multiplied by KPROP, the proportional constant, to generate ICMD, the current command. Proportional velocity loop is selected when the PROP switch is on. PROP is turned off at power-up.

Proportional velocity loops are much easier to stabilize than integrating loops, so they are often used during machine setup. However, they also allow steady-state velocity error and therefore, they are generally replaced with integrating loops when the machine is fully operational.

8.9.2.2 Integrating Velocity Loop

If an integrating velocity loop is selected, then the velocity error is integrated and multiplied by KVI, the velocity integration constant. Velocity feedback is subtracted from this signal, then the signal is multiplied by KV, the velocity loop gain, to form ICMD. This velocity loop is selected when PROP is off.

8.9.3 Torque Command

In a few applications, the VECTOR**STAR** is given a "torque" command. Actually, this is a current command, but at lower speeds, motor torque is approximately proportional to current. In this case, VCMD is multiplied by KPROP to form ICMD. Note that this differs from the proportional velocity loop only in that VFB is not subtracted from VCMD. The switch TQ must be on to select the torque mode and off for all other modes. The position loop should be off (PL off) when the VECTOR**STAR** is running in Torque command mode. The VECTOR**STAR** will turn PL off when TQ is turned on.

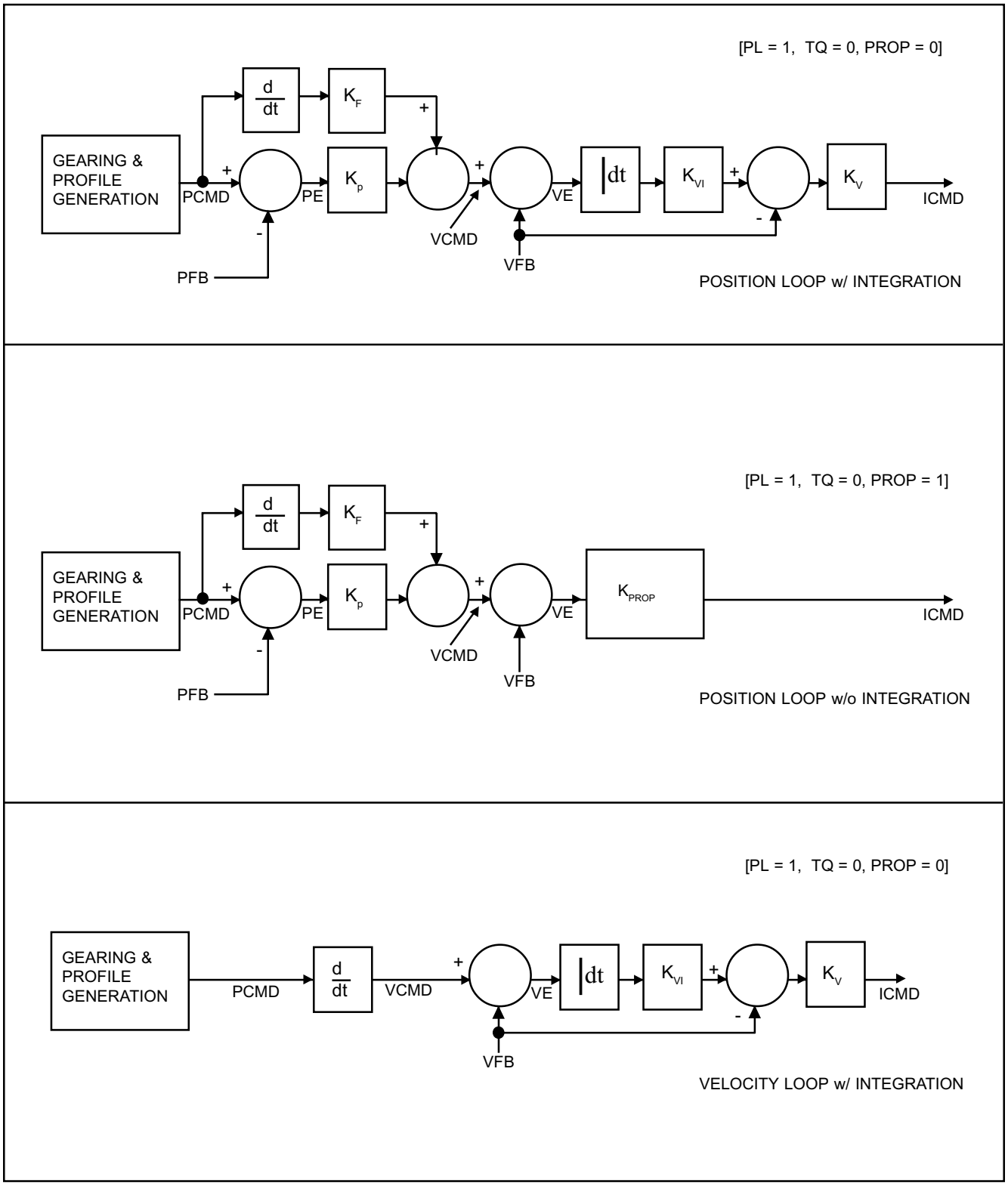
When TQ is turned on, PL is forced off

8.9.4 Power-Up Control Loops

The VECTOR**STAR** has, at power-up, the following settings:

- Position loop enabled or disabled (PL on or off) depends on the last setting used. PL is remembered through power-down.
- No feed-forward (KF=0)
- Integrating Velocity Loop (PROP off, TQ off)

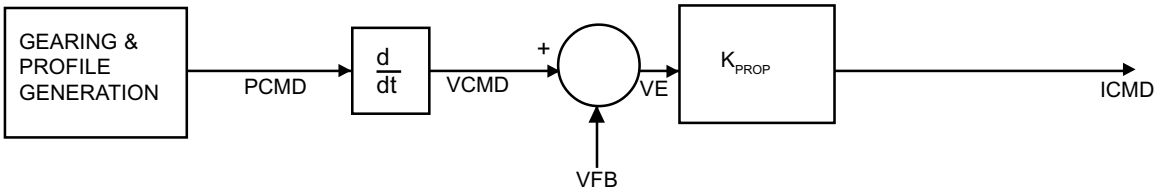
These settings meet the requirements of a large number of applications. Figure 8.9 shows each of the five VECTOR**STAR** controller modes.



*Figures are correct when PDF = 1

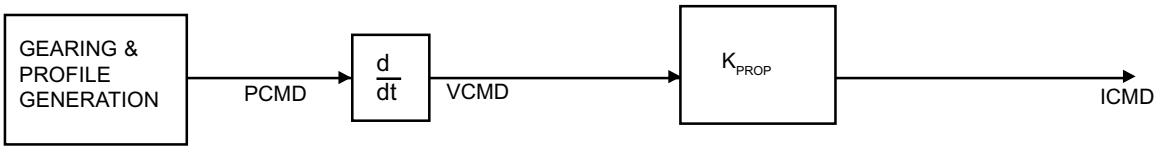
Figure 8.9. VECTORSTAR Control Modes

[PL = 1, TQ = 0, PROP = 1]



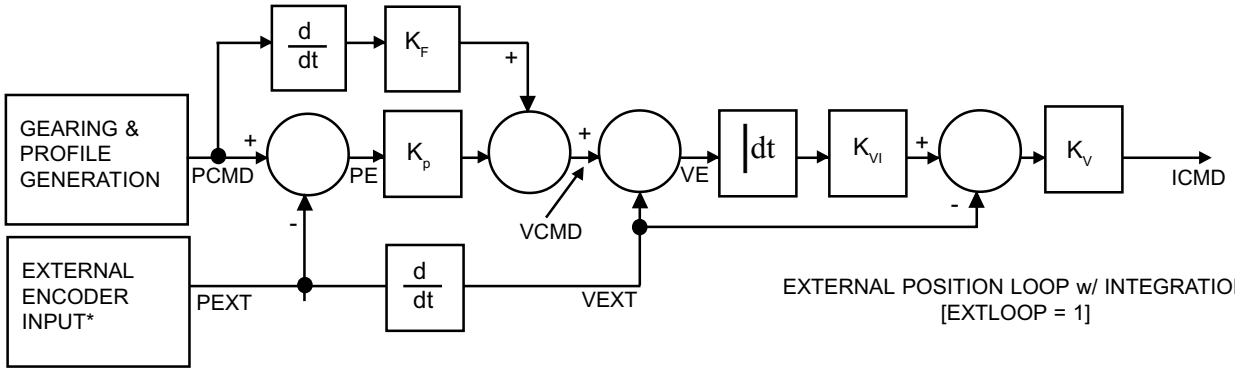
VELOCITY LOOP w/o INTEGRATION

[PL = 1, TQ = 0, PROP = 1]



OPEN LOOP - TORQUE (CURRENT)

[PL = 1, TQ = 0, PROP = 0]



EXTERNAL POSITION LOOP w/ INTEGRATION
[EXTLOOP = 1]

*Encoder resolution must be equal to RDRES (4096 counts standard) and must be mounted directly on to motor.

CHAPTER 9
SPINDLE PROGRAMMING

9.1 INTRODUCTION

Induction motors produce torque from the mutual interaction between the rotor current and the stator magnetization current. The rotor current of induction motors is induced from the stator current. The objective of vector control is to supply stator current so that its two current components, the rotor current and the magnetizing current, are exactly supplied to the motor as commanded from a controller. The vector control condition can be met if the controller is programmed to read the exact orientation of the rotor magnetic axis, which can be estimated via the speed detector and the amount of slip frequency.

9.2 SYSTEM COMPENSATION VARIABLES

The following variables for system compensation are unique for the induction motor. All are set at the factory.

IND — This switch must be ON (IND=1) to configure your VECTOR**STAR** for an induction motor.

IMAG — Magnetization Current. Unit is in percent of IMAX. Normally, this current should be close to the no-load current of the motor at rated voltage and base frequency. The optimal IMAG value should be set at the factory. Magnetization current is now reduced automatically during field weakening operation. Magnetization current set by the "IMAG" variable is maintained at speeds below base speed. At speeds higher than the base speed, magnetization current will be inversely proportional to the speed. Base speed is set at the factory and is generally determined by the start of the constant horsepower point of the curve.

VBASE — Base speed RPM. Field weakening starts here.

BSLIP — Expected Slip frequency, when IBASE current is supplied to the motor. BSLIP = 524 corresponds to 1 Hz of slip frequency.

IBASE — Stator Current that produces BSLIP at VBASE. Unit is in percent of IMAX.

SLOPE — When in field weakening region, slip per current must be increased.

VADVTBL — This is the maximum speed of the slip table. Set the same as VMAX if possible.

SLIP — Not programmable. Current Slip is displayed using this command. In this representation, 1 Hz slip corresponds to SLIP = 524. To obtain optimal parameters for a specified motor, the two following

variables help determine the optimal operating conditions with a dyno setup. Note that the two variables below are for factory use only.

MADV Manual Slip Control Switch MADV = 1 for Manual Slip Control. Default = 0, factory use only.

MSLIP When MADV = 1, amount of slip is solely controlled by MSLIP. Factory use only. In this representation, 1 Hz slip corresponds to SLIP = 524.

9.3 COMMANDS FOR INDUCTION DRIVE

The following variables are related to the vector control:

SLIPLIM User variable SLIPLIM limits the maximum amount of slip. At all speeds, slip frequency cannot exceed SLIPLIM value. If 1 Hz slip is maximum, put SLIPLIM = 4294967, which is 65.54 x 65536. Normally the maximum slip is 4-8 times greater than rated slip at base speed.

TROTOR This variable sets the rotor time constant (75 corresponds to 300 mSec).

GIMAGF Low pass filter gain for dynamic Imag control.

PDF System uses PDF velocity control by using KPROP and KVI only. PDF = 0 for most spindle applications. KV is not used when PDF = 0.

KPROP Determines the proportional gain (Range: 1 - 32000).

KVI Determines integral frequency (Range: 0 - 4095).

PROP If integral control is undesirable, you may set PROP to 1 to disable integral action (proportional gain only).

VXAVG Averaging of VEXT.

LOAD User's application code may be embedded in the firmware and loaded upon power-up. A switch variable LOAD enables (=1) or disables (=0) this loading upon power-up. Note that it is remembered through powering down.

Enable/disable loading of user application code from PROM.

9.4 DEDICATED VARIABLES

The following 16 user variables are for the application software use only. These variables are similar to X-variables except that they have dedicated names.

VZR	Zero speed in RPM
VUP	Up to speed in RPM
VUPH	Up to speed hysteresis in percentage
GEARIC	GEARI Value for C-axis (range of -32768 to 32767)
GEAROC	GEARO Value for ratio for C-axis (range of 0 to 32767)
MTIMER	Delay timer for motor over temperature (long integer)
DTIMER	Delay timer for drive over temperature (long integer)
GEARIO	GEARI value for open mode (range of -32768 to 32767)
GEAROO	GEARO value for open mode (range of 0 to 32767)
P1, P2	Orient positions in PRD Unit (range of 0 to 4095)
VORNT	Orient velocity in RPM (long integer)
MORNT	Orient mode (range of 1 - 5)
VXNUMO	VXNUM value for open mode (long integer)
VXDENO	VXDEN value for open mode (long integer)
VXNUMC	VXNUM value for C-axis mode (long integer)
VXDENC	VXDEN value for C-axis mode (long integer)

9.5 SPINDLE FUNCTIONS

9.5.1 Power Meter Output

VECTOR**STAR** generates the output which is proportional to the torque. This signal can be used to drive a power meter with 6.67 volts representing 100% power.

9.5.2 General Purpose Input/Output Pin Assignments

The VECTOR**STAR** has 16 inputs and 8 outputs of General Purpose I/O pinouts. These can be used for spindle functions such as C-axis Command, Zero Speed output, etc. If you are using the spindle function provided by our user program, you must assign the general purpose I/O.

9.5.3 Application Code

A user application code can be embedded into the PROM for permanent storage. At power-up, the VECTOR**STAR** will load this code to the RAM and execute it immediately if LOAD is 1.

Autobaud must be disabled on power-up so that the application software can be executed without waiting for the establishment of communications, which is expected by the CNC.

If the you want to run a program other than the embedded one, enter LOAD = 0 and load the user program. The new program will reside in RAM and be ready to run. At power-up, the new program will run rather than the embedded program.

9.6 TYPICAL APPLICATION PROGRAM FOR VECTORSTAR

This section provides details for a user program for a typical spindle application. If you think the program is suitable to your application, contact the factory for embedding the code into EPROM.

The following application software implements general spindle drive functions which may interface with a higher level controller through the GENERAL PURPOSE I/O connectors provided by the VECTOR**STAR**.

Note that this example may NOT work for your applications. Very often, modifications are necessary under different circumstances.

The user variables X1 to X55 and XS1 to XS15 are reserved by the program.

9.6.1 Features

This application software is able to implement the following features:

1. Zero Speed
2. At Speed
3. C-Axis (Spindle Axis) / Open Loop Modes
4. Drive Over Temperature
5. Motor Over Temperature
6. Orient
7. Spindle Reset
8. Drive Enable
9. Drive Ready
10. Remote

Each function is described below.

9.6.2 VECTORSTAR General Purpose Input/Output Definition

In this application software, the VECTOR**STAR** GENERAL PURPOSE I/O connectors are used as an interface to a higher level controller and from sensors. Three inputs are commands from the controller, two inputs are the sensor signals, and seven outputs are for feedback signals for VECTOR**STAR** status to the controller. Connect these I/Os and program the higher controller properly based on the following definitions to implement the spindle functions:

A. Input to VECTORSTAR

I1 Spindle Axis/Open Loop (1/0)
I2 Orient Request (1)
I3 Dual Orient Position Select (select P1 or P2 position)
I4 Drive Over-Temperature (0) (Negative Logic, 1 is normal)
I5 Motor Over-Temperature (0) (Negative Logic, 1 is normal)
I6 Spindle Reset (1) (data reset)
I7 Drive Enable (1)

B. Output from VECTORSTAR

O1 Zero Speed (1)
O2 At Speed (1)
O3 Orient Complete (1)
O4 Drive Ready (1)
O5 Drive Over-Temperature (0) (Negative Logic, 1 is normal)
O6 Motor Over-Temperature (0) (Negative Logic, 1 is normal)
O7 C-Axis (Spindle Axis) Active

A "1" following the I/O name indicates positive logic. For example when O1 is "1," Zero Speed is indicated. A "0" following the I/O name indicates negative logic. For example, O5 being "0" indicates over-temperature.

C. REMOTE input

The Pulse Enable input can be the dedicated REMOTE input located on connector C2, pin 9.

9.6.3 Software Function Description

Zero Speed
The feature Zero Speed controls an output signal that indicates the motor is stopped. The output is energized when the absolute value of the measured motor speed is at or below the parameter value named VZR in RPM units. The measured motor speed is stored in the non-programmable variable VAVG. This feature is active in all modes of VECTOR**STAR** operation.

The Zero Speed output is the General Purpose O1 output.

Variables used:

VAVG — Non-programmable variable updated by VECTOR**STAR.**

VZR — Defined zero speed upper range by the user, in RPM. The software reads the speed as zero if VAVG is less than VZR.

At Speed
The feature At Speed controls the GP4 output signal that indicates the motor is rotating within a speedband around the commanded speed. The absolute value of the differential between measured speed and command speed is compared to the At Speed parameter named VUP. When the absolute value of the differential value is less than VUP, the At Speed output energizes. After the At Speed output energizes, a second hysteresis parameter named VUPH is added to VUP to provide a new band around the command speed. The output will remain energized until the command speed or the measured speed **is changed significantly enough** so that their absolute differential is greater than the VUP+VUPH band. Once the output goes off, the absolute differential between measured speed and command speed must be less than the VUP parameter for the output to reenergize.

The measured speed, command speed, VUP parameter, and VUPH parameter are in percentages. The measured speed is stored in the non-programmable variable VAVG.

The VECTOR**STAR** will accept a +/- analog DC voltage, over the range of +10 to -10 VDC, as a speed command input in the Spindle Axis (i.e. Closed Loop) mode or the Open Loop mode. The VECTOR**STAR** converts the analog voltage input into a bi-directional speed command. The value of the speed command is controlled by the parameters GEARI and GEARO.

Review the **Spindle Axis/Open Loop Modes** below for a discussion on the GEAR and GEAR parameters. The design output channel for At Speed is O2.

Variables used:

VXAVG, VAVG — Non-programmable variables from The feature Zero Speed controls an output signal that when on indicates the motor is stopped. The output is energized when the absolute value of the measured motor speed is at or below the parameter value named VZR in RPM units.

VUP — Defined band around the speed command, in RPM.

VUPH — A second hysteresis parameter to provide a new band around the speed command after the motor speed entered into the first band, in percentage of RPM.

Before using this function, make sure VXNUM and VXDEN are correct for the current gear ratio.

Spindle Axis (C-Axis) / Open Loop Modes
The Spindle Sxis mode, also called C-axis mode, is used by the higher level controller to control both the angular position and the speed of the motor. The higher level controller can use the motor position information to close the loop. The spindle motor is used as a servo motor in this mode. This feature develops a technique known as “rigid tapping,” allowing operators more precise depth and torque control during tapping.

The velocity command can be scaled by the gear ratio, i.e. GEARI and GEARO. In the Open Loop mode, GEARI and GEARO are programmed by the variables GEARIO and GEAROO. While in the spindle axis mode, GEARIC and GEAROC are used for the new gear ratio.

Once the maximum RPM for the Open Loop mode is specified, the GEARI and GEARO values can be calculated. Load the required GEARI value in GEARIO and the GEARO value in GEAROO.

The VECTOR**STAR** will accept a +/- analog DC voltage, over the range of +10 to -10 VDC, as a speed command input in the Spindle Axis (i.e. Closed Loop) mode or the Open Loop mode. The VECTOR**STAR** converts the

analog voltage input into a bi-directional speed command. When the I1 input is on, the Spindle Axis mode is selected, and when the I1 input is off, the Open Loop mode is selected.

When the Spindle Axis mode is selected, the VECTOR**STAR** turns on the I 7 output named the Spindle Axis Active output.

The Spindle Axis mode is used by the controller to control the angular position of the motor as well as the speed of the motor. Positional feedback is provided to the controller.

The command voltage is scaled to a given speed by the programmable variables GEARI and GEARO.

The RPMmax for the Open Loop mode is 10000 RPM. The Spindle Axis mode can have a RPMmax of 2000 or less. The controller and VECTOR**STAR** must be matched such that the maximum command voltage from controller to VECTOR**STAR** represents the same maximum RPM.

The designed input channel for C-Axis is I1.

Variables used:

GEARIO GEARI	For Open Loop Mode. The spindle command speed is scaled by GEARIO and GEAROO.
GEAROO GEARO	For Open Loop Mode
GEARIC GEARI	For Spindle Axis Mode. The Axis speed command is scaled by GEARIC and GEAROC.
GEAROC GEARO	For Open Loop Mode

The ranges of the above four variables are the same as GEARI (-32768 to 32767) and GEARO (0 to 32767).

VXDENC	VXDEN for Spindle Axis Mode
VXNUMC	VXNUM for Spindle Axis Mode.
VXDENO	VXDEN for Open Loop Mode.

VXDENC and VXNUMC are used to convert the internal value to RPM for the specific gear ratio, so Zero Speed and At Speed work in both modes.

VXNUMO	VXNUM for Open Loop Mode
X47	Open Loop KPROP value
X48	Open Loop KVI value
X49	Open Loop LPFHZ value
X50	C-axis Loop KPROP value
X51	C-axis Loop KVI value
X52	C-axis Loop LPFHZ value

VXDENO and VXNUMO are used to convert the internal value to RPM for a specific gear ratio, so Zero Speed and At Speed can work in both modes.

Orient

The VECTOR**STAR** assumes speed and position control of the motor during Orient and rotates the spindle motor to a desired programmable position. An Orient cycle is initiated by the Orient Request I2 input. After the motor is oriented, the VECTOR**STAR** energizes the Orient Complete O3 output. The Orient Request I2 input must remain on to complete the Orient and to hold the motor in the Orient position. After the Orient Request I2 input is turned off, the VECTOR**STAR** decodes the state of the Spindle Axis/Open Loop I1 input and assumes the state of the selected mode.

The Orient Request is received from GENERAL PURPOSE connector, I2. The software is so designed that I2 must be maintained in order to complete the Orient and hold the motor in the Orient position.

Orient Request I2 must be turned off when the Drive Ready O4 output is turned off, because the VECTOR-**STAR** will inhibit the Drive Ready output from turning on if the Orient Request input is on.

After the Orient Request has been removed from I2 (I2 = 0), the user must issue the Drive Enable again so that the software can go back to the state of Spindle Axis or Open Loop defined by I1.

Input I2 must be turned off when Drive Ready O4 is off.

The orient positional units are in R/D (resolver to digital) bits/rev. The VECTOR**STAR** has 4096 bits in l revolution of the motor. The orient position is stored in the variable named **P1**. The range of values for P1 is 0 to 4095. To capture the P1 orient position, place the motor in Tram mode (i.e. disable the motor such that the motor can be rotated by hand). Position the motor to the orient position. Using the VECTOR**STAR** communication device, perform the command P PRD. The number

returned is the number that is loaded in terminal mode into P1 representing the orient position.

9.6.3.1 1:1 Application

The VECTOR**STAR** can be used in an application in which one revolution of the motor results in one revolution of the spindle (1:1 application). When the VECTOR**STAR** receives an Orient Request I2 input, the externally commanded speed is captured. The VECTOR**STAR** then assumes control of the motor, maintaining the externally captured command speed. The Orient command then brings the motor to a complete stop at a random position prior to moving the motor to the orient position in less than one revolution. The speed at which the motor moves from the random stopped position to the orient position is controlled by the parameter named VORNT, in units of RPM. Various modes of operation can be selected in the 1:1 application as to how the motor is moved to the orient position after stopping.

The modes of operation are selected by a parameter named MORNT and are as follows.

MORNT = 0. The motor will orient in the direction it was rotating when the Orient Request input activated.

MORNT = 1. The motor will orient in the CCW direction. The direction of rotation is that direction as seen from the non-shaft end of the motor.

MORNT = 2. The motor will orient in the CW direction. The direction of rotation is that direction as seen from the non-shaft end of the motor.

MORNT = 3. NOTE: This mode selects the Orient cycle designed for the 2:1 application described in the next section.

MORNT ³ 4. The motor will orient in the shortest direction.

Variables used:

MTIMER - Motor OT (i.e. Over-Temperature) timer. The time from when the Motor OT I5 input thermostat contact opens until the drive faults, terminating application software execution and disabling the motor. The time allows the controller to stop the motor before the VECTOR**STAR** stops the motor. Care must be taken to select a value that is low enough to prevent damage to the motor. The units are in milliseconds.

P1 (PORNT no longer used) - Orient position in R/D units. There are 4096 R/D units per revolution of the motor, so the programmed value can be in the range of 0 to 4095.

VORNT - Orient speed of the motor is in RPM units. During the Orient process the motor will be stopped and then commanded to move to the Orient position in less than 1 motor revolution at this specified velocity. In the 2:1 application (discussed below), VORNT also is the speed at which the motor rotates to locate the HOME input.

VUP - Defined band around the speed command, in RPM.

Associated with the Orient software is a test that verifies the motor is stopped followed by a test that verifies the motor is at the orient position. The in-position orient test is executed regularly while the motor is oriented. If the test fails, the motor will be disabled and application software execution will be halted. The application software can be restarted with a power reset of the VECTOR**STAR** or by toggling the Spindle Reset GP6 input on. The **Drive Ready** section later in this chapter describes the conditions that will allow the VECTOR**STAR** to enable after a fault. The Orient software tests have the following parameters that must be programmed.

X41 - The X41 parameter is added to and subtracted from the P1 position, providing a pass/fail band for the orient in-position test. If the measured PRD motor position is outside the band established by X41, the test will fail. X41 is in R/D units and there are 4096 units per motor revolution. The value of X41 is determined by how far the motor can be from the orient position without causing a machine problem.

X42 - The test that senses the motor has stopped compares a new motor position with a previously measured motor position. When the absolute difference between those values is less than the value in X42 for a given number of consecutive test loops, the VECTOR**STAR** concludes the motor has stopped and the in-position test can begin. X42 is in R/D units and there are 4096 units per motor revolution. The value of X42 must be greater than 0 and a value that will sense the motor has stopped. Typically allow for a minimum of an I R/D unit change, so use a minimum of X42 = 2.

X43 - The maximum loop count of the motor stopped software test. This parameter prevents the software from staying in the motor stopped test loop if the Orient fails. If the loop count were to exceed X43, the VECTOR**STAR** would be disabled and application software execution would be halted. Initially, set the value high and orient several times. After each orient, monitor X36 to capture the number of actual loops of the test. Select an X43 value based on X36 that will provide a buffer which will prevent nuisance failures.

X45 - The test that senses the motor has stopped compares a new motor position with a previously measured motor position. When the absolute difference between those values is less than the value in X42 for a given number of consecutive test loops as defined by X45, the VECTOR**STAR** concludes the motor has stopped and the in-position test can begin. The orient in-position test must not start until the motor has stopped. X45 defines the number of times the motor position must be captured and not change outside the X42 parameter window. A typical value is less than 5.

9.6.3.2 2:1 Application

The VECTOR**STAR** can be used in an application in which two revolutions of the motor result in one revolution of the spindle (i.e. 2:1 application). In this application the speed of the motor will be twice the speed of the spindle, and the feedback resolution of the spindle will be twice that of the motor with the feedback transducer remaining in the motor. One concern is that the motor orients on the correct revolution so that the spindle is aligned in the correct position. If the motor was oriented on the other revolution, the spindle would be 180 degrees from the orient position. When the VECTOR**STAR** receives an Orient Request I2 input, the externally commanded speed is captured. The VECTOR**STAR** then assumes control of the motor, maintaining the externally captured command speed. The motor is then commanded to rotate in the direction the motor was rotating when the Orient Request was received, and at the speed defined by the parameter named VORNT. The VECTOR**STAR** then monitors the dedicated HOME input. The input must go off or be off and then on at which time the VECTOR**STAR** brings the motor to a stop at a random motor position prior to moving the motor to the orient position in less than one revolution.

The X46 parameter defines the amount of time in milliseconds the VECTOR**STAR** will allow to sense the HOME input running or being off and then turning on. If the X46 time frame is exceeded, the VECTOR**STAR** will stop the motor, disable the drive, and halt application software execution. The Orient position is defined by the parameter P1 and is in R/D units with a range of 0 to 4095. The speed at which the motor moves to the orient position from the random stopped position is controlled by the parameter VORNT, in RPM. VORNT thus controls the speed of the motor in searching for the HOME input and also in determining the speed of the motor during the Orient move from the random stopped motor position to the orient position. The 2:1 application is selected by making MORNT equal to 3. The spindle has a cam that will activate the HOME input over the half of the spindle revolution in which the orient position exists. The orient position must be centered on the cam and the cam can not exceed one-half of the spindle revolution or there is a chance the spindle could be oriented 180 degrees from the desired position. After the motor is moved to the orient position, the motor stopped test is executed followed by the orient in-position test. When both tests are complete, the VECTOR**STAR** energizes the Orient Complete O3 output if the HOME input is active. The orient in-position test will continue to be executed while the Orient Request GP2 input is on.

While the motor is oriented, all inputs are monitored. The Zero Speed feature is active and the Zero Speed O1 output will be on. The At Speed feature is active and the At Speed O2 output will be controlled by the value of the analog command voltage. Since the motor is being maintained at the Orient position, a command voltage of 0 will activate the At Speed output. The Over-Temperature inputs are included in the monitoring. The Spindle Reset I6 input will not have any affect.

After Orient is complete, command P PRD can be used to check the Orient results. The designed input channel for Orient is I2.

Variables used:

P1, P2 Desired orient positions in R/D units. VECTOR**STAR** currently uses 12-bit mode and these are per resolution. The programmable Orient position can be assigned from 0 to 4095.

VORNT Desired motor orient speed in RPM. The motor will move to the P1 or P2 in less than 1 revolution at this speed. If the motor moves only a short distance, the VORNT may not be reached before the stop at the Orient position.

MORNT Orient mode selection.
- 0: from the random stopped position in the direction the motor was rotating when the Orient Request input was received.
- 1: same as mode 0 but in CCW direction (see from non-shaft end of motor).
- 2: same as mode 0 but in CW direction.
- 4: same as mode 0 but in the shorter direction.

X41 A pass/fail band for the orient in-position test.

X42 Must be greater than 0 and is used to sense the motor has stopped by comparing X42 with the movement of the motor within one sampling period.

X43 A maximum loop count of the software test for motor being stopped. If the test has been done more than X43 times, the VECTOR**STAR** will conclude the test has failed.

X45 Defines the number of times the motor position must be captured and not change outside the X42 parameter window. Normally set to 3.

All X-variables are in PRD units (4090 counts per revolution for 12-bit resolution of R/D).

Motor Over-Temperature
The VECTOR**STAR** senses a motor over-temperature (OT) condition and reports this condition by energizing the Motor OT I6 output. It senses the OT condition when the Motor OT I5 input goes to a logical 0 (turns off). The VECTOR**STAR** also starts a programmable timer. When this timer is exceeded with the OT condition present, the current application job will be terminated and the motor will be disabled. The Motor OT timer is programmed by the variable MTIMER in milliseconds. Normally, MTIMER should not be greater than 120 minutes. The designed input channel for Motor Over-Temperature is I5.

To re-enable application software execution, the following conditions must be true.

1. No OT condition.
2. Orient Request I2 input off.
3. Toggle the Spindle Reset I6 input on.

While the OT condition exists and the OT timer is active, normal VECTOR**STAR** operation continues.

If the OT condition is removed prior to the termination of the timer, the Motor OT output is turned off and VECTOR**STAR** operation continues as normal.

Variables Used:

MTIMER Used for the delayed termination of the software execution after the motor over-temperature condition has been sensed. MTIMER is in milliseconds.

TMR2 Motor Over-Temperature timer. The time from when the Motor OT I5 input thermostat contact opens until the VECTOR**STAR** faults, terminating application software execution and disabling the motor. The timer is initialized by the parameter MTIMER in units of milliseconds. The TMR2 timer will decrement from the initialized MTIMER value to 0.

Drive Over-Temperature
The VECTOR**STAR** senses an internal over-temperature condition and reports this condition by energizing the Drive OT I4 output. The VECTOR**STAR** also starts a programmable timer that, when exceeded with the OT condition present, will terminate application software execution and disable the motor. To re-enable application software execution, the following conditions must be true.

1. No OT condition.
2. Orient Request I2 input off.
3. Toggle the Spindle Reset I6 input on.

While the OT condition exists and the OT timer is active, normal VECTOR**STAR** operation continues. If the OT condition is removed prior to the termination of the timer, the Drive OT output is turned off and VECTOR**STAR** operation continues as normal.

Variables Used:

DT

DTIMER Used for the delayed termination of the software execution after the drive over-temperature condition has been sensed.

DTIMER is in milliseconds, and a maximum of 10 to 15 minutes is recommended.

TMR1 Drive Over-Temperature timer. The time from when the Drive OT I4 input thermostat contact opens until the VECTOR**STAR** faults, terminating application software execution and disabling the motor. The timer is initialized by the parameter DTIMER in units of milliseconds. The TMR1 timer will decrement from the initialized DTIMER value to 0.

Spindle Reset
The Spindle Reset I6 input is used to reset the VECTOR**STAR** application software execution after a fault has been detected. The VECTOR**STAR** will halt application software, disable the motor, and remove the Drive Ready O4 output after a fault has been detected. To start application software execution, the following conditions must occur.

1. The Spindle Reset I6 input must be toggled off to on.
2. The Drive OT I4 input must be in a non-fault state or on.
3. The Motor OT I5 input must be in a non-fault state or on.
4. The Orient Request I2 input must be off.

A Spindle Reset will only start application software execution. The Drive Ready O4 output will then turn on when the REMOTE input is on, the Drive Enable O7 input is on, and the Orient Request I2 output is off.

If an OT fault terminated application software execution, the Spindle Reset I6 input must be toggled off to on to turn off the OT output if the OT fault condition has been removed (i.e. the OT input has turned on).

Toggling the Spindle Reset I6 input with the application software executing will have no effect on the VECTOR**STAR**.

In the user program, an alarm label, A$, is used to monitor the RESET input. See Chapter 10, "User Programs," for more information about alarm function programming.

Drive Enable
The Drive Enable I7 input works in conjunction with the Pulse Enable dedicated REMOTE VECTOR**STAR** input to energize the motor and place the motor under VECTOR**STAR** control. Both inputs must be on to enable the VECTOR**STAR** and to turn on the Drive Ready O4 output. If the Drive Enable I7 input is lost with the motor rotating, the motor will be commanded to 0 speed. After the motor reaches 0 speed, the VECTOR**STAR** will be disabled, which de-energizes the motor and turns off the Drive Ready O4 output. If the Pulse Enable input is turned off, the VECTOR**STAR** is disabled and the motor will be de-energized regardless of the state of the Drive Enable I7 input. If the motor is rotating when the Pulse Enable input is lost, the motor will coast.

The Drive Enable I7 input and Pulse Enable input being off is not a fault state. The VECTOR**STAR** application software is executing, which includes monitoring all VECTOR**STAR** inputs. This is comparable to a fault state in which a Spindle Reset I6 input is required to start program execution.

When Drive Enable is turned on with the application software executing, the Pulse Enable input on, and the Orient Request input off, the VECTOR**STAR** will enable and turn on the Drive Ready O4 output.

Drive Ready
The Drive Ready output is the O4 output. The Drive Ready output will be on when the VECTOR**STAR** is in a state to accept and act on a speed command input voltage or an Orient Request input.

The VECTOR**STAR** is powered on with the Drive Ready output off. The following states must be true to turn the Drive Ready output on.

1. Pulse Enable on. (REMOTE SIGNAL)
2. Drive Enable on.
3. Orient Request off.
4. The VECTOR**STAR** application software is executing. Application software begins executing at power-up. The software execution could be terminated by several conditions such as a Motor or Drive OT condition that has exceeded the allocated time defined by MTIMER for the motor and DTIMER for the drive.

The Drive Ready O4 output will turn off with any of the following conditions.

1. Pulse Enable off.
2. Drive Enable off.
3. A Motor or Drive OT condition has timed out which terminates application software execution.
4. An Orient failure which terminates application software execution.
5. A VECTOR**STAR** firmware sensed fault that terminates application software execution.

If the Drive Ready output is turned off due to the Pulse Enable input or the Drive Enable input being turned off, then turning both back on with the application software executing will turn on Drive Ready as long as the Orient Request input is off. If Drive Ready is turned off due to termination of application software execution, or the software is halted while Drive Ready is off, a Spindle Reset I6 input will be required to start software execution. Drive Ready will then turn on after the application software has started when Pulse Enable is on, Drive Enable is on, and Orient Request is off.

Pulse Enable
The Pulse Enable (i.e. REMOTE) VECTOR**STAR** input works in conjunction with the Drive Enable I7 input to energize the motor and place the motor under VECTOR**STAR** control. Both inputs must be on to enable the VECTOR**STAR**. If the Pulse Enable input is turned off, the VECTOR**STAR** is disabled and the motor will be de-energized. If the motor is rotating when the Pulse Enable input is lost, the motor will coast. If both Pulse Enable and Drive Enable I7 inputs are lost, the motor will be de-energized, and if the motor was rotating, it will coast.

9.6.4 Units Review

9.6.4.1 Gear Ratio

If you can use the on-board A/D converter, for 10 volt input generating 10000 RPM, GEARI/GEARO = 700/16384, and based on this, 70/16384 for 1000 RPM, accordingly, for 10 volt input.

9.6.4.2 VXAVG

To obtain a correct VXAVG for the different gear ratios, the VXNUM and VXDEN should be calculated by the equation:

$$VXNUM / VXDEN = (104700/1) \times (10000/V_{max})$$

Where V_{max} is in RPM and 4096 counts per revolution for 12-bit R/D mode is assumed. The gear ratio is indirectly represented by V_{max} in the equation. Please see Chapter 10, "User Programs," for more information.

Again using the example of 10000 RPM as a maximum input speed command,

$$VXNUM/VXDEN = 104700/1$$

to obtain the correct reading of VXAVG in PRM with the gear ratio = 700/16384.

9.6.5 TL Sheet

The VECTOR**STAR** is shipped with a normal TL sheet (factory settable parameters) and a list of all the programmable variables. All the parameters mentioned in this chapter are set to the default values by the factory and are included in the TL sheet.

9.6.6 Software Listing

The user should be familiar with the VECTOR**STAR** system and how to program it before attempting to read the code listing of the user program or designing a new program for the VECTOR**STAR**.

The following listing is similar though not identical to the one embedded on the EPROM.

```
;THIS ALARM IS TO SENSE THE RESET
INPUT (I6)

A$ I6 1
? O5 EQ 1 ? I4 EQ 1 O5 0
? O6 EQ 1 ? I5 EQ 1 O6 0
? (1-O4)&I4&I5&(1-I2) EQ 1 RUN 9
? I4 EQ 0 O5 1
? I5 EQ 0 O6 1
END

;INITIALIZATION OF THE SYSTEM
POWER-UP$
LPF 1

9$
PL 0
RAMP 1
LPF 1
GEARI=GEARIO
GEARO=GEAROO
```

```
VXNUM=VXNUMO
VXDEN=VXDENO
O7 0
GEAR 1
X2=24;NORMAL INPUT

;PROCESS THE FAULTS AND READ THE
INPUT

10$
? O5 EQ 1 GOSUB 50          ;GO FOR
                            ;PROCESSING
                            ;DRIVE OT
? O6 EQ 1 GOSUB 60          ;GO FOR
                            ;PROCESSING
                            ;MOTOR OT
X5=IN
? X5 NE X2 GOSUB 20         ;GO FOR
                            ;PROCESSING
                            ;THE INPUT
? READY EQ 0 GOSUB 90       ;DRIVE NOT
                            ;ENABLED
? REMOTE&I7 EQ 0 ? READY EQ 1 GOSUB
100                          ;REMOTE IS
                            ;OFF

;ZERO SPEED

X4=VAVG
? X4 LT 0 X4=0-X4
? VZR LT 0 VZR=5
IF X4 LE VZR
O1 1
ELSE
O1 0
ENDIF

;AT SPEED
? VUP LT 0 VUP=10
? VUPH LT 0 VUPH=10
X10=VXAVG-VAVG              ;ERROR
                            ;BETWEEN THE
                            ;ACTUAL
                            ;SPEED AND
                            ;COMMAND
? X10 LT 0 X10=0-X10;ABSOLUTE
                            ;VALUE
IF X10 LT VUP               ;WITHIN THE
                            ;BAND?
O2 1
ELSE
IF X10 GT VUP+VUPH;OUT OF 2ND
                  ;BAND?
O2 0
ENDIF
ENDIF
GOTO 10                     ;REPEAT LOOP

;INPUT PROCESS
20$
X6=X5!X2
X7=X5&X2
X2=X5
X5=X6-X7
? X5&8 EQ 8 GOSUB 50        ;DRIVE OT
? X5&16 EQ 16 GOSUB 60      ;MOTOR OT
? X5&1 EQ 1 GOSUB 30        ;SPINDLE AXIS
? X5&2 EQ 2 GOSUB 40        ;ORIENT
RET

;SPINDLE AXIS

30$
GEAR 0
IF I1 EQ 1
  GEARI=GEARIC              ;SWITCH THE
                            ;GEAR RATIO
  GEARO=GEAROC
  VXNUM=VXNUMC              ;SWITCH THE
                            ;SCALE TO
 VXDEN=VXDENC
   O7 1
   RAMP 0
ELSE                        ;BACK TO
                            ;OPEN LOOP
                            ;MODE
   GEARI=GEARIO
   GEARO=GEAROO
   VXNUM=VXNUMO
   VXDEN=VXDENO
   RAMP 1
   O7 0
ENDIF
? O3 EQ 0 GEAR 1
RET

;SPINDLE ORIENT

40$
? I2 EQ 0 RET
? READY EQ 0 RET
O7 OFF
CONTINUE                    ;TURN GEAR
                            ;OFF

J 0
W 0
ZPE
PL 1
MRD P1 VORNT                ;USE MRD TO
                            ;ORIENT
```

```
W 0
O1 1
O3 1
O2 0
TIL I2 EQ 0
O3 0                    ;HOLD THE
                        ;POSITION
                        ;UNTIL  I2 = 0
PL 0
RET

;DRIVE OVER-TEMPERATURE

50$
IF I4 EQ 0
? O5 EQ 0 TMR1=DTIMER
O5 1
? TMR1 EQ 0 GOTO 80
ELSE
O5 0
TMR1=0
X2=X2!8
ENDIF
RET

;MOTOR OVER-TEMPERATURE

60$
IF I5 EQ 0
    ? O6 EQ 0 TMR2=MTIMER
    O6 1
    ? TMR2 EQ 0 GOTO 80
    ELSE
        O6 0
        TMR2=0
        X2=X2!16
ENDIF
RET

;APPLICATION FAULT, EXIT EXECUTION

80$
O4 0
CONTINUE
? REMOTE EQ 1 ? READY EQ 1 J 0
W 0
DIS
O7 0
O3 0
O2 0
PL 0
END

;ENABLE THE DRIVE WITH REMOTE=1,
READY=1 AND WITHOUT ORIENT

90$
? I4&(1-O5) EQ 0 GOSUB 50
? I5&(1-O6) EQ 0 GOSUB 60
? REMOTE&I7&(1-I2) EQ 0 RET
X2=X2&65533
GEAR 1
EN
O4 1
RET

;REMOTE=0 OR DRIVE ENABLE=0 AFTER
THE DRIVE WAS ENABLED

100$
IF I7 EQ 0
CONTINUE
? REMOTE EQ 1 J 0
W 0
DIS
O4 0
O3 0
PL 0
ELIF REMOTE EQ 0
O4 0
O3 0
PL 0
GEAR 0
ENDIF
? O4 EQ 0 ? REMOTE EQ 0 GOSUB 110
RET

;WAIT FOR REMOTE=1

110$
? I4&(1-O5) EQ 0 GOSUB 50
? I5&(1-O6) EQ 0 GOSUB 60
? I7 EQ 0 DIS
? I7 EQ 0 RET
? REMOTE&I7&(1-I2) EQ 0 GOTO 110
X2=X2&65533
GEAR 1
O4 1
XS1 0
RET

;DO THESE AFTER A VFS5 ERROR OCCURS

ERROR$
O2 0
O7 0
O4 0
O3 0
```

```
PL 0
GEAR 0
DIS
END
```

9.6.7 A Sample Application Program for the VECTORSTAR

As mentioned earlier, the VECTOR**STAR** can have a set of spindle functions built in the drive which meets higher controller or CNC requirements. The interface follows general input and output lines of the drive. But if you wish to design a program to implement simple functions, simply type LOAD = 0 and load your program down to the VECTOR**STAR**. Your program will be executed either by issuing the RUN command from MOTION LINK, or automatically at power-up by using the $POWER-UP label in your program.

The following sample program will activate the machine using HOME input and CYCLE input through the C2 connector on the front of the VECTOR**STAR**.
First, this application checks to see if the system is ready to be enabled, then looks at REMOTE input, which is the external control for the power stage. If REMOTE is 1, the drive will be enabled and wait for either HOME or CYCLE input (1 is active, but both cannot be 1 at the same time).

When HOME input is 1, the motor will run at a constant speed at X1 [RPM]. Assign X1 before running this program. The motor can be stopped by turning HOME off (reset to zero).

If CYCLE input is 1, the motor will receive the velocity command from the analog input through the electrical gear function (you must set gear ratio GEARI and GEARO before running the program. For instance, GEARI = 560 and GEARO = 16384 for 10-volt input = 8000 RPM). So you can control motor speed by controlling analog input voltage.

You can switch from analog input to a constant speed, by turning CYCLE off and HOME on.

Note that all the variables stored in the VECTOR**STAR** will be remembered through power-down.

PROGRAM LIST

```
POWER-UP$           ;START PROGRAM
                    ;HERE AT POWER UP
GEAR 0              ;NOT RECEIVING
                    ;VELOCITY COMMAND
                    ;FROM ANALOG
                    ;INPUT
ABAUD 0             ;NO AUTO BAUD

TIL OK2EN EQ 1      ;WAIT FOR SYSTEM
                    ;READY

5$
TIL REMOTE EQ 1     ;WAIT FOR REMOTE
                    ;INPUT IS 1
EN                  ;ENABLE THE DRIVE
10$
IF REMOTE&HOME EQ 1
;CHECK BOTH REMOTE AND HOME ARE 1
GEAR 0              ;NOT RECEIVING
                    ;VELOCITY COMMAND
                    ;FROM ANALOG INPUT
J X1                ;RUN A CONSTANT
                    ;SPEED AS X1, WHICH
                    ;HAS BEEN GIVEN BY
                    ;USER (MAY BE FROM
                    ;LAST OPERATION)
TIL HOME EQ 0       ;POLLING THE HOME
                    ;INPUT
J 0                 ;STOP THE MOTOR IF
                    ;HOME=0
W 0                 ;WAIT FOR ZERO
                    ;SPEED
D 500               ;WAIT HALF SECOND
                    ;FOR MOTOR STOP
                    ;COMPLETELY
ENDIF

30$
IF REMOTE&CYCLE EQ 1
;WAIT UNTIL BOTH REMOTE AND CYCLE
;INPUT ARE 1

GEAR 1              ;READY TO RECEIVE
                    ;VELOCITY COMMAND
                    ;FROM ANALOG INPUT
ENDIF
TIL CYCLE EQ 0
CONTINUE
J 0
W 0
D 500

GOTO 10             ;LOOP BACK FROM
                    ;10$

ERROR$              ;START THIS ROUTINE
                    ;ONLY IF AN ERROR
                    ;OCCURS
```

```
IF LSTERR EQ 14      ;IF THE ERROR IS
                     "POWER BUS", WAIT
                     ;UNTIL SYSTEM IS
                     ;READY
TIL OK2EN EQ 1
RUN 5          ;IF SYSTEM IS OK, RUN
                     ;FROM 5$
ENDIF
B                    ;OTHERWISE, QUIT
```

KOLLMORGEN
Motion Technologies Group

TL VSA12/2031E12
ISSUE 1 SH 1 OF 5
WRITTEN BY T. CONNER 6/4/97

TEST LIMITS AND MODIFICATION DATA
VSA-COMP1 FORM REV:B

MOTOR DATA:

MODEL	V-2031EN	
MAXIMUM SPEED	12000	RPM
CONTINUOUS TORQUE	35	LB.FT.
ROTOR INERTIA	0.014	LB.FT.SEC.^{2}
LOAD INERTIA RANGE	0.015	LB.FT.SEC.^{2}
BASE SPEED	1500	RPM

AMPLIFIER DATA:

MODEL	VSA-12		
TACH MONITOR	1750	RPM/VOLT	C2-PIN2
CURRENT MONITOR	16	AMPS RMS/VOLT	C2-PIN 4
CONTINUOUS CURRENT	40	AMPS RMS/PHASE OR	50%
PEAK CURRENT	80	AMPS RMS/PHASE OR	100%
NOM. SYSTEM VOLTS	230	VOLTS RMS	
MAX. SYSTEM SPEED	12000	RPM AT NOM. SYSTEM VOLTS	

AMPLIFIER COMPENSATION:

CURRENT LOOP

A-PHASE	R7	499K	C8	3300pf	C9	OPEN
B-PHASE	R4	499K	C5	3300pf	C6	OPEN
C-PHASE	R1	499K	C2	3300pf	C3	OPEN

The following variables are listed in the UNPROTECTED VARIABLES section.

POSITION LOOP
- PL = SOFWARE SWITCH
- KP = GAIN
- KF = FEED-FORWARD GAIN

VELOCITY LOOP
- KPROP = PROPORTIONAL GAIN
- KVI = INTEGRATING GAIN
- PROP = SET TO 1 TO DISABLE INTEGRAL ACTION

LOW-PASS FILTERS
- LPF = SOFTWARE SWITCH
- LPFHZ = HERTZ

SYSTEM SPEED
- VOSPD = OVERSPEED is a user changeable variable up to 120% of VMAX.
- VMAX = MAXIMUM SYSTEM SPEED. This is a PROTECTED VARIABLE.

COPY CODE

ECN:

TL VSA12/2031E12
ISSUE 1 SH 2 OF 5

TEST LIMITS AND MODIFICATION DATA
VSA-COMP1 FORM REV:B

RESOLVER CODING - RESOLVER RESOLUTION 12 BIT

Label VSA-COMP 1, in the box provided, with amplifier current rating and motor compensation.

EXAMPLE:

MODEL:
12/2031E12
REV. NO 1

ADDENDUM:

Attached to this TL SHEET, please find a listing of VSA variables.

```
;VS1\12-0012 V4.0
;ISSUE 1
;MOTOR = V-2031EN
;
;AMPS    = 40
;VOLTS   = 230
;
;UNITS:
;
PNUM    = 1
PDEN    = 1
VNUM    = 44739
VDEN    = 10
ANUM    = 4474
ADEN    = 1000
INUM    = 4095
IDEN    = 100
PXNUM   = 1
PXDEN   = 1
VXNUM   = 87250
VXDEN   = 1
;
;
;UNPROTECTED VARIABLES:
;
A2D     = 0
ABAUD   = 0
AMAX    = 100000
ACC     = 100000
ADDR    = 0
BAUD    = 9600
DEC     = 4500
DEP     = 0
DIR     = 1
DTIMER  = 15000
ECHO    = 1
GEARI   = 840
GEARIC  = 140
GEARIO  = 840
GEARO   = 16384
GEAROC  = 16384
GEAROO  = 16384
GIMAGF  =10
ILIM    = 100
IMAG    = 10
KC      = 200
KF      = 0
KP      = 2000
KPROP   = 2000
KV      = 7705
KVI     = 1000
LOAD    = 0
LPF     = 1
LPFHZ   = 100
MORNT   =3
MSG     = 1
MTIMER  = 120000
PECLAMP = 100
PEMAX   = 32767
PDF     = 0
```

```
PL     = 0
PLIM     = 0
PMAX     = 2000000000
PMIN     = -2000000000
P1          = 2144
P2          = 0
PROTARY  = 409600000
REGKHZ   = 1000
ROTARY   = 1
SCRV     = 1
SLIPLIM  = 45000000
TRIP        = 0
VDEFAULT = 200
VOFF     = 0
VORNT    = 250
VOSPD    = 13200
VUP      = 15
VUPH     = 5
VXDENC   =1
VXDENO   = 1
VXNUMC   = 523500
VXNUMO   = 87250
VZR      = 5
WTIME    = 1000
X41      = 20
X42      = 2
X43      = 30
X45      = 3
X46      = 5000
X47      = 2000
X48      = 1000
X49      = 100
X50      = 4000
X51      =1000
X52      =100
Z0          = 1
Z1          = 0
Z10         = 5
Z11      = 37
```

```
;TL VS1/12-0012 V4.0
;ISSUE 1
MOTOR    = V-2031EN
;
;AMPS     = 40
;VOLTS    = 230
;
;
;UNITS
;
INUM     = 4095
IDEN     = 100
VNUM     = 44739
VDEN     = 10
;
;
;TUNING VARIABLES:
;
KF          = 0
KP          = 1000
KPROP    = 2000
KVI         =500

;PROTECTED VARIABLES:
;
ANGLD    = 0
BSLIP    = 3500
FOLDD    = 100
FOLDR    = 900
FOLDT    = 200
ICONT    = 50
IMAX     = 100
IND         = 1
IZERO    = 25
MADV     = 0
MANG     = 165
MSLIP    = 0
POLES    = 512
SGOOSE   = 0
SLOPE    = 30
VADVTBL = 12500
VBASE    = 2500
VMAX     = 12000
Z20         = 15000
```

CHAPTER 10
USER PROGRAMS

10.1 INTRODUCTION

The information in this chapter will enable you to understand the capabilities of the system. You will also explore important considerations that must be addressed before you implement your own application. Examples of programming techniques will aid you in developing your own applications.

10.2 PROGRAMMING TECHNIQUES

User programs are combinations of the commands stored in the VECTOR**STAR** memory. These programs are stored in non-volatile RAM; they are not lost when the VECTOR**STAR** is powered down. User programs are composed mainly of the commands that have been described in earlier chapters. In addition, there are commands necessary for controlling the way the program executes; these commands are covered in this chapter. The first section describes the VECTOR**STAR** Editor, which allows you to enter and modify programs from the terminal. If you have not already done so, read Chapter 8 before proceeding. This manual is written to be read sequentially. Do not attempt to save time by skipping ahead to this chapter.

READ THIS ENTIRE SECTION CAREFULLY.

This section discusses programming practices. The VECTOR**STAR** has a flexible language. You must follow proper programming principles to ensure that the flexibility does not lead to overly-complex programs. If you follow good programming practices you will:

- be able to modify programs when the application changes;
- have fewer programming errors;
- have an easier time fixing the programming errors that do occur; and
- be able to get help with errors you cannot fix.

People who are new to programming often disregard good programming practices because they have not experienced the problems that result from poor programming practices. Save yourself the misery of having to re-write your entire program. Follow these steps:

Always hardwire personal safety functions. Never program these functions.

1. DO NOT PROGRAM SAFETY FUNCTIONS.

   Always hardwire safety functions. This includes EMERGENCY STOP or ESTOP. You should not depend on your program for safety functions because of three potential problems: 1) You can easily make programming errors (software problem); 2) A function on the VECTOR**STAR** may not work in exactly the way you expect it to in every condition (firmware problem); and 3) A critical component in your system may fail and prevent the function from working (hardware problem). Remember, safety functions are rarely exercised so that if one of these problems does occur, it can go undetected indefinitely. If personal safety is involved, always hardwire the function.

Programming errors can damage your equipment. Use caution when programming equipment protection functions.

2. USE CAUTION WHEN PROGRAMMING EQUIPMENT PROTECTION FUNCTIONS.

   Sometimes you can hardwire equipment protection functions, but other times this is impractical and you must program the functions. If this is the case, be very careful. Remember, if your program has an error, it can result in damage to your equipment. For example, suppose you want to wire your motor thermostat so that when a fault occurs, the present machine cycle continues until complete. In this case, you must program the function (hardwiring the thermostat would result in motion stopping the moment a thermostat fault is encountered). Carefully test these functions.

3. WRITE A SIMPLE SPECIFICATION FOR YOUR APPLICATION.

   Write an outline of all the functions your application will require before you start programming. This will serve as a specification. Everyone who is involved with your system (customers, supervisors, co-workers, operators) should agree on the specification. While last-minute requests for program changes will still occur, this is a reasonable step towards reducing the incidence of such requests.

4. WRITE A FLOWCHART OF YOUR PROGRAM.

   People new to programming often have a natural distaste for writing flowcharts. Many view flowcharts as something between a crutch and unnecessary work. Most experienced programmers have a different view. The most important point about flowcharts is that they are virtually required if you need help over the telephone. Always write flowcharts for programs that are longer than 20 to 30 lines.

5. COMMENT YOUR PROGRAM.

   Always comment your programs. Comments help explain your program to other people. Keep in mind that others may need to modify your program in the future. Comments also help you remember why you chose certain ways to do things.

6. AVOID SPAGHETTI CODE.

   A program with too much branching is often called spaghetti code because of the look of the flowcharts. Avoid a lot of branching, especially branching up (towards the top of your program); logic in programs that branch down is more intuitive and thus, less prone to errors. If you do branch up, branch to the top of a major section. In most programs there should only be one or two places that you branch up to. Feel free to use small loops (2 or 3 lines) which, of course, repeatedly branch to the top of the loop. Avoid branching between sections.

7. AVOID USER SWITCHES THAT MODIFY BLOCKS OF CODE.

   Switches that modify functions can be difficult to understand. This is commonly done when programmers attempt to use one block of code for two similar functions. If possible, write two different blocks of code rather than trying to use one block for two functions.

10.2.1 Example Application

Suppose you are working on a project that is defined by someone besides yourself. It may be a co-worker, a supervisor, a customer, or an operator. For this example, we will use a customer. Suppose you have this conversation:

Customer: "My machine feeds plastic from a roll onto a conveyor then cuts it into sheets. The length of the sheet varies. There is a registration mark on each plastic sheet which is detected while the plastic is moving. After this mark is detected, the motor must move the plastic a variable distance and stop. There is a stop input that should stop and disable the VECTOR**STAR** after it completes the cycle."

You: "Are there other parameters that should be variable such as speed, acceleration, and deceleration?"

Customer: "Now that you mention it, all those parameters should be variable. I also need an output at the end of the move to start the saw blade rotating."

You: "How often do these variables change?"

Customer: "About once or twice a year."

You: "Do you mind typing them in from a keyboard?"

Customer: "No. That would be fine."

You: "What controls the start of the move?"

Customer: "My PLC activates an input. Can ESTOP be programmed so that it can be overridden when the cycle is almost complete?"

You: "No. Since ESTOP is a safety function, it is always hardwired to remove power."

10.2.2 Application Specification

1. Allow a variable cut length, acceleration, deceleration, and speed. Use user variables X1-X4 as follows:

   X1 Acceleration
   X2 Deceleration
   X3 Speed
   X4 Cut Length (added to registration mark)

2. Turn on an output at the end of the move. This output will be connected to start the saw. Use output O1.

3. Allow contacts that stop the process after the present cycle is complete. Use input I1.

4. Wait for a start signal to begin each cycle. Use input I2.

10.2.3 Application Flowchart

When you write flowcharts use three symbols: a circle, a square, and a diamond. A circle indicates the start or end of a program. It also indicates the start or end of a subroutine. A square is an execution block. That is, the VECTOR**STAR** should do something: turn on an output, print a message, or command motion. A diamond is a decision block. There are two exits from a diamond: one if the condition is true and the other if it is false. Figure 10.1 presents a sample flowchart for this application.

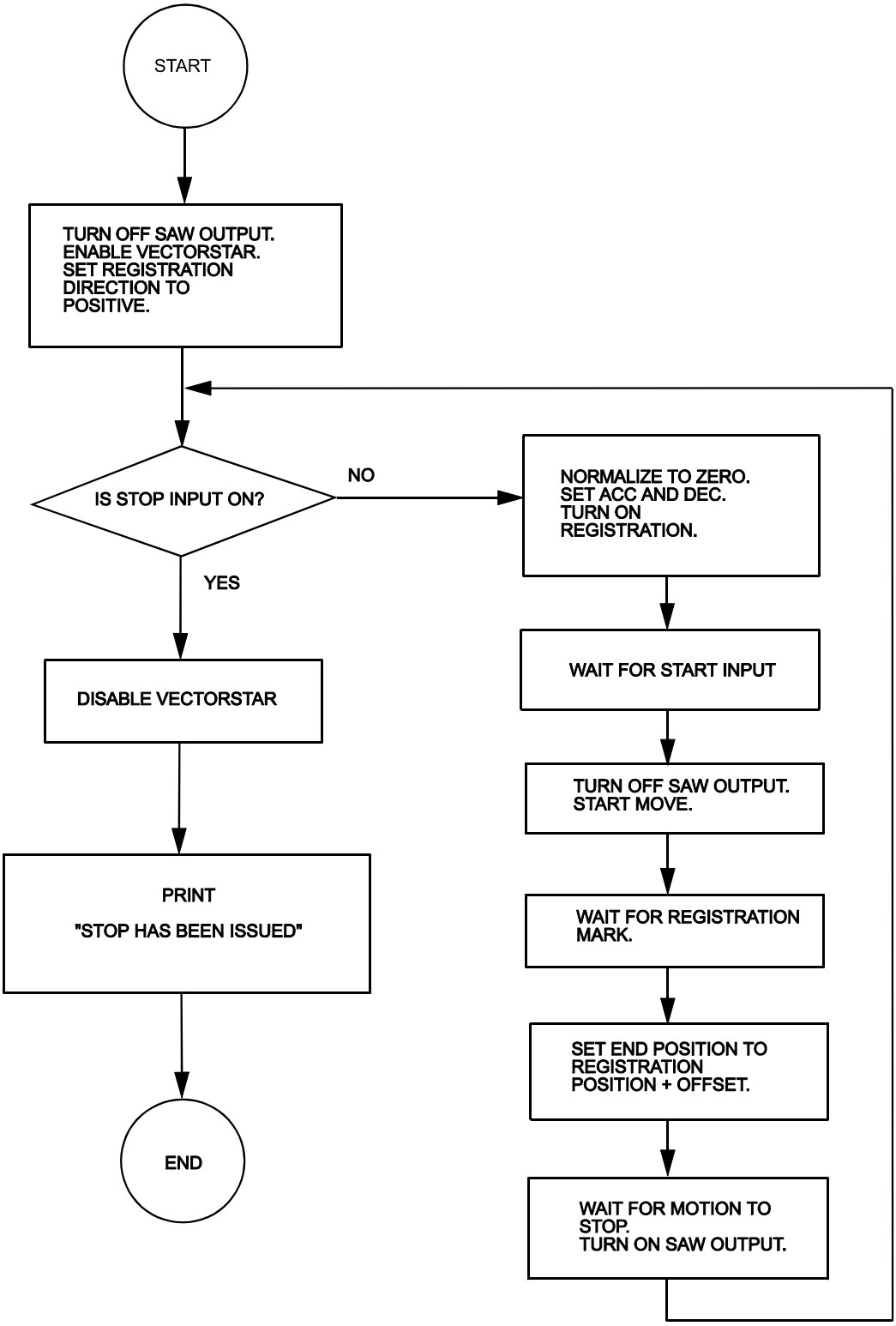


Figure 10.1 Sample Flowchart

10.2.4 Commented Program

The following program will work for this application:

```
; THIS PROGRAM DOES NOT HAVE A
; HEADER (THAT IS, THE BEGINNING
; SECTION WHICH HAS A LOT OF
; INFORMATION). USE THE PROGRAM IN
; APPENDIX E AS AN EXAMPLE OF A
; PROGRAM HEADER.

1$                  ;START OF PROGRAM
O1  OFF             ;TURN OFF THE SAW
                    ;OUTPUT
CAPDIR = 1          ;SET REGISTRATION
                    ;DIR. POSITIVE
EN                  ;ENABLE
                    ;VECTORSTAR

5$                  ;BEGIN LOOP
IF  I1  EQ  1       ;IS STOP INPUT ON?
 GOTO  10           ;GOTO TO "STOP
                    ;ROUTINE"
ELSE
 NORM  0            ;NORMALIZE TO 0
 ACC = X1           ;SET ACC
 DEC = X2           ;SET DEC
 CAP  ON            ;TURN ON CAPTURE
 TIL  I2  EQ  1     ;WAIT FOR START
                    ;INPUT
 O1  OFF            ;TURN OFF SAW
                    ;OUTPUT
 J  X3              ;START MOVE
 TIL  CAP  EQ  0    ;WAIT FOR
                    ;REGISTRATION
 JT  PCAP+X4  0     ;SET END POSITION
                    ;TO CAPTURED
                    ;POSITION PLUS AN
                    ;OFFSET (X4)
 TIL  SEG  EQ  0    ;WAIT FOR MOTION
                    ;TO STOP
 O1  ON             ;TURN ON SAW
                    ;OUTPUT
 GOTO  5            ;GO TO TOP OF LOOP
ENDIF

10$                 ;START OF "STOP
                    ;ROUTINE"
K                   ;DISABLE
                    VECTORSTAR
P  "STOP HAS BEEN ISSUED"
B                   ;STOP EXECUTION
```

10.2.5 Customer Service

If you need help with software or understanding VECTOR**STAR** functions, you can contact the Regional Kollmorgen Sales Office (See Appendix D). Ask for the Sales Applications Engineer. Please observe the following procedure:

1. Contact Kollmorgen for each new problem. Occasionally, an applications sales representative may refer you to the Engineering Department. However, if you call later with a new problem, please ask for an applications sales representative.

2. Be prepared to provide the following items:

   a. A written spec of the system;

   b. A flowchart; and

   c. A hard copy of the program.

3. Be prepared to take the following actions, should the applications sales representative determine that these actions are necessary:

   a. Strip out sections of your program to help locate a problem.

   b. Rewrite sections of your program that do not conform to the programming practices described in this chapter.

   c. Video tape your machine to help demonstrate the problem.

If you need help with your program, Kollmorgen is committed to helping you. VECTOR**STAR** software support is provided by:

1. Helping you organize your program.

2. Explaining proper programming practices.

3. Discussing VECTOR**STAR** functions.

Contact the local Industrial Drives sales application representative. All Regional Sales Offices are listed in Appendix D of this manual.

10.3 EDITING

Writing or modifying a program is called editing. If you are using MotionLink Plus, you can skip this section. There are two ways you can edit a VECTOR**STAR** program. The VECTOR**STAR** has a simple resident editor. As an alternative, you can edit your program on a computer and transmit it to the VECTOR**STAR**. Motion Link is a software package designed specifically for this purpose. Motion Link runs on IBM-PC's and compatibles, and it handles the communications between the VECTOR**STAR** and the computer. Motion Link also features a full-screen editor.

Editing with Motion Link is preferred because it has more features than the resident editor, and it allows you to save your program on disk. Having the program on disk is a significant advantage since it is a simple matter to transmit, or download, the program should the VECTOR**STAR** be replaced or multiple systems be programmed.

10.3.1 Motion Link Editor

Chapter 8 provides an in-depth procedure for installing and using Motion Link. This section provides you with enough information to get started in most cases. Enter a simple program with the following procedure:

1. Establish communications with the VECTOR**STAR** as discussed in Chapter 7.

2. Press the right arrow key to display the menu bar. Select PROGRAM.

3. Select NEW.

4. Enter this program:

```
10$
    P "HELLO WORLD"
    B
```

5. Press the escape key to exit the Motion Link Editor.

6. Follow the instructions on your computer screen. Motion Link will ask you if you want to save your program. Enter "Y" and give the name "TEST" as the name of your program.

7. Motion Link will now ask you if you want to transmit the program to the VECTOR**STAR**. Enter "Y."

8. After the transmission is complete, you should receive the interactive prompt (—>). Type:

```
RUN 10
```

Your program should print:

```
HELLO WORLD
—>
```

This should provide you with enough information to enter the examples from this chapter. Read Chapter 8 for a complete description of Motion Link.

10.3.2 VECTORSTAR Resident Editor

If you are not using the VECTOR**STAR** Editor, skip ahead to the next section, "Building a Program." The VECTOR**STAR** resident editor allows you to enter small programs and make changes without Motion Link. Note that you can use this editor from Motion Link just as you would use it from a terminal.

To enter the VECTOR**STAR** Editor, type:

```
ED
```

When you are in the Editor, the VECTOR**STAR** will respond with the editor prompt:

```
E->
```

To exit the Editor, press the escape key.

10.3.2.1 Editor Print (P)

The Print (P) command prints a program line (or lines) then goes to that line. Each line in the program memory has a number. Many editor instructions, such as Print, expect you to specify the line number (or numbers) that applies to the instruction.
Type in the following example from the VECTOR**STAR** Editor:

```
P BEG END
```

The VECTOR**STAR** will print the entire program and go to the end of the program. When you specify a range, the command works for all the lines in the range. You can specify one line. For example, type:

```
P 1
```

The VECTOR**STAR** will print and go to line 1. If you want to print the current line, then do not specify a line. For example, to print the current line, type:

```
P
```

If you attempt to print a line that is not in the program, such as line 100 of a 10-line program, the Editor will issue an error such as:

```
BAD ENTRY
```

10.3.2.2 Next Line

If you enter an empty line, then the VECTOR**STAR** goes down one line in the program and prints that line. The empty line is entered by pressing the enter key. This makes it easy to move down through the program.

10.3.2.3 Password (PASS)

The VECTOR**STAR** Editor has password protection. The password allows you to prevent the user program from being changed. If the password is set, the program cannot be changed, but it can still be displayed. The password can be up to six characters long. The default setting of the password is null (i.e. empty), which means there is no password protection. From the Editor, type:

```
PASS
```

The VECTOR**STAR** will ask you for the new password.

If you do not want password protection, enter an empty line. Note that the NEW command, discussed later, also clears the password.

10.3.2.4 Insert (I)

Entering the Insert (I) command causes the Editor to enter the Insert mode. When you are in the Insert mode, everything you type is put directly into the program memory. Exit the Insert mode by pressing the escape key or entering an empty line. For example, type:

```
P  BEG      ;GO TO THE BEGINNING OF
            ;THE PROGRAM
I           ;ENTER THE INSERT MODE
```

The VECTOR**STAR** will respond with:

```
I->
```

indicating that you are in the Insert mode. Now type:

```
;TEST LINE FOR LEARNING ABOUT THE
;EDITOR
```

Press the escape key to exit the insert mode. Type:

```
P  1
```

The VECTOR**STAR** should respond with:

```
1     ;TEST LINE FOR LEARNING ABOUT
      ;THE EDITOR
```

You can specify the line you want to insert directly. For example:

```
I  5
```

enters the Insert mode. The next line you type is entered directly into the program as the new line 5. Subsequent lines, 6, 7, and so on, follow line 5.

10.3.2.5 Find (F)

The Find (F) command will search down through the program memory for a particular word, letter, or string of characters. For example, the Find command can be used to find the word “EDITOR” from the Insert command above. From the Editor, type:

```
P BEG       ;USE P TO GO TO TOP AND
            ;SEARCH
F
```

The VECTOR**STAR** should respond with:

```
FIND WHAT?
F->
```

then type:

```
EDITOR
```

The Find command will find line 1 since the word EDITOR occurs in that line. Now F can be used to find the next line with “EDITOR.” Type:

```
F
```

and the VECTOR**STAR** should respond with:

```
FIND WHAT?  EDITOR?
F->
```

In this example, the Find command has a default "FIND WHAT" string. The default is the find string from the last Find command. If you enter an empty line, the next line with "EDITOR" will be found. If you do not want to use the default string from the last Find command, type in the word or words you want to find this time. Pressing the escape key will abort the F command.

If the Editor cannot find the specified word, it will respond with "NOT FOUND" and return to the edit mode:

```
NOT FOUND
E->
```

10.3.2.6 Change (C)

The Change (C) command is similar to the Find command. However, Change allows you to change the string you found. Also, Change only searches the current line. Use the P command to go to line 1 and print the line you typed from the previous discussion of the Insert command. Type:

```
P 1
```

The VECTOR**STAR** should respond:

```
1       ;TEST LINE FOR LEARNING ABOUT
        ;THE EDITOR
```

Now use the C command to change "EDITOR" to "VECTOR**STAR** EDITOR." Type:

```
C
```

The VECTOR**STAR** will respond with:

```
CHANGE WHAT?  "EDITOR"?
C->
```

Again, "EDITOR" from the Find command is the default input. Press the return key to accept the default and the VECTOR**STAR** will respond with:

```
CHANGE TO WHAT?
C->
```

Now type:

```
VECTORSTAR  EDITOR
```

The VECTOR**STAR** will change the line to read:

```
1       ;TEST LINE FOR LEARNING ABOUT
        ;THE VECTORSTAR EDITOR
```

C has defaults for both the "CHANGE WHAT" and the "CHANGE TO WHAT." This allows you to step through memory, changing each occurrence of one string to another string with minimal keystrokes. Like F, pressing the escape key will abort the process and return to the Edit mode.

10.3.2.7 Delete (DEL)

The Delete (DEL) command can be used to delete one line or a whole range of lines.

DO NOT TYPE THESE EXAMPLES!

For example:

```
DEL  5  10      ;DELETE LINES
                ;5,6,7,8,9, AND 10
DEL  12         ;DELETE LINE 12
DEL             ;DELETE CURRENT
                ;LINE
```

All of these delete instructions are valid.

For an example that you can type in, if you entered line 1 "TEST LINE FOR LEARNING ABOUT THE EDITOR," then type in the following command to delete that line:

```
P  1            ;POINT AT THE FIRST LINE
                ;"TEST LINE..."
DEL  1          ;DELETE THAT LINE
```

Line 1 should be deleted.

10.3.2.8 Size

The VECTOR**STAR** program memory has space for about 16000 characters. If you want to see how much memory is left, type:

```
SIZE
```

The VECTOR**STAR** will respond with:

```
65 % LEFT
```

which means the available space is about 65%. If you try to enter a program larger than the VECTOR**STAR** can store, an error will be generated.

10.3.2.9 NEW

The NEW command resets the password and clears the program. The user program is stored in battery backed-up RAM. Normally, the program is remembered indefinitely. However, if power to the VECTOR**STAR** is lost when it is executing an Editor command, there is a small chance that the program will be corrupted. This can happen, for example, if power is lost during the Change or Delete command. In this case, the VECTOR**STAR** will generate a "USER PROGRAM CORRUPT" error and the program cannot be modified or run. If this happens, use the NEW command to clear the user program and reset the corrupt error.

DO NOT TYPE IN THIS EXAMPLE!

```
NEW
```

The NEW command also clears the editor password.

The >BDS command, discussed in Section 10.10, will also reset the program so that it is no longer corrupt, although it will not clear the password.

10.4 BUILDING A PROGRAM

Programs are sequences of commands, most of which can also be executed directly from the keyboard. A program stores the sequences of these *normal* commands. Examples of these commands are MI, MA, and P (Print). However, in order for a program to run properly, other commands, called *program control commands*, are required. Examples of these commands are GOTO and GOSUB.

10.4.1 Basic Commands

10.4.1.1 Labels

Labels are used to mark places in the program where execution begins or continues. There are two kinds of labels: general purpose labels and dedicated labels.

General purpose labels are numbers from 0 to 500 followed by a dollar sign ($). You can execute a program that begins at a general purpose label with the RUN command. You can jump to a label from within your program with the GOTO and GOSUB commands. RUN, GOTO, and GOSUB are described later in this chapter.

Dedicated labels each have specific functions. Dedicated labels include alarms, auto programs, and the user error handler. These labels are *letters* or *words* followed by a dollar sign. For example, A$ is the A-Alarm label. Dedicated labels cannot be used by the RUN, GOTO, or GOSUB commands. These labels are discussed with multi-tasking later in this chapter.

10.4.1.2 RUN

The RUN command is used to start the program from the Interactive mode. For example, type:

```
RUN 3
```

If there are no errors, and if label 3 is in the user program, then program execution begins at label 3. The RUN command can execute all valid general purpose labels. If the label is not in the program, an error is generated and no part of the program is executed. You cannot use the RUN command for dedicated labels.

Before the program is run, the VECTOR**STAR** searches the entire program for some types of errors. If, after you enter a RUN command, an error is detected, the VECTOR**STAR** will display the appropriate error message together with the offending line. Also, RUN verifies that the program has not changed since the last edit. If the program has changed, a "PROGRAM CORRUPT" error is generated. The program corrupt error can be cleared, though this requires that the entire program be erased with the Editor NEW command or the >BDS command. If a "Program Corrupt" error occurs, and it was not caused by losing power while you were editing, this may indicate a serious condition. Contact the factory.

10.4.1.3 Break (B)

The Break (B) command is the opposite of RUN; it stops program execution and normally returns to the interactive state. The Break command does not stop motion. Profile commands are allowed to continue until they are complete. If you want to break the program and stop motion, precede the Break command with the Stop (S) command.

10.4.1.4 GOTO

The GOTO command is used within the program to jump to a label. Before the following example of GOTO can be done, another function of the Print (P) command should be explained. From the terminal, type:

```
P "THIS PRINTS TEXT JUST LIKE I TYPED IT IN"
```

and the result will be:

```
THIS PRINTS TEXT JUST LIKE I TYPED IT IN
```

When using the Print command, characters between double quotes are printed back without modification.

Returning to the GOTO command, use the Editor Insert command to enter the following short program:

```
2$
P "AT LABEL 2"
GOTO 3
P "NEVER GOT HERE"
3$
P "AT LABEL 3"
B
```

Exit the Editor and type:

```
RUN 2
```

The result should be:

```
AT LABEL 2
AT LABEL 3
```

Avoid the use of GOTO commands in favor of Block-IF, Quick-IF, and GOSUB commands. This practice makes programs more readable and easier to modify.

10.4.1.5 GOSUB and RET

The third command that uses labels is GOSUB. The GOSUB command goes to a subroutine at the specified label. For example:

```
GOSUB 66
```

begins a subroutine at label 66. The RET command returns from the subroutine and begins executing the program one line below the original GOSUB command.

GOSUB's can be nested up to four levels.

For example, type in the following program:

```
4$
GOSUB 5
P "RETURNED FROM SUBROUTINE 5"
B
;
5$
P "EXECUTING SUBROUTINE 5"
RET
```

Exit the Editor and type:

```
RUN 4
```

The result should be:

```
EXECUTING SUBROUTINE 5
RETURNED FROM SUBROUTINE 5
```

10.4.2 CONDITIONAL COMMANDS

The VECTOR**STAR** provides several conditional commands that allow your program to make decisions. Conditional commands include ? (Quick-IF), TIL, IF, and ELIF. These commands all depend on *conditions*. A condition is an arithmetic comparison of any two numbers, variables, or expressions. The VECTOR**STAR** supports all 6 common types of arithmetic conditions. Note that you should not use the =, >, or < symbols for these conditions. Instead, you must use the following two-character codes:

Table 10.1 VECTORSTAR Conditions

GT	Greater Than
GE	Greater Than Or Equal To
LT	Less Than
LE	Less Than Or Equal To
EQ	Equal To
NE	Not Equal To

10.4.2.1 QUICK IF (?) Command

The ?, or Quick-IF, is a single-line command that allows you to specify a condition, a command to be executed if the condition is true, and another to be executed if the condition is false. The format of the ? command is:

? condition TRUE-command : FALSE-command

TRUE-command is executed if the condition is true and FALSE-command is executed if the condition is false.

Both TRUE-command and FALSE-command are optional, although at least one must be present.

Some examples of the ? command are:

```
? X1 GT 5 P "X1 > 5" : P "X1 <= 5"
? VFB GT 3000 P "HIGH SPEED" : P
"LOW SPEED"
? 2*X2-5 LE X1/100 GOSUB 40
? X1/2*2 EQ X1 GOTO 5
                    ;GOTO 5 IF X1 IS
                    ;EVEN. DO NOTHING
                    ;IF X1 IS ODD.
? I4 EQ 1 J 2000    ;I4 IS A JOG BUTTON
```

Note that each condition has an exact opposite: EQ & NE, LE & GT, and LT & GE are all pairs of opposites. Since the ? command allows both TRUE-command and FALSE-command, you have your choice of which command to use in the condition. For example, the two ? commands that follow have exactly the same effect:

```
? X1 EQ 10 B : P "X1 OK"
                    ;BREAK IF X1 > 10
? X1 NE 10 P "X1 OK" : B
                    ;BREAK IF X1 > 10
```

The ? command can be used to make a loop counter. Suppose you want to go to subroutine 25 twenty times. You could just write *GOSUB 25* twenty times, but it would probably be better to use a *program loop*. The following statements show how the ? command can be used to control that program loop:

```
X30 = 1             ;X30 IS THE LOOP
                    ;COUNTER
12$                 ;THE LOOP BEGINS
                    ;AT 12$
GOSUB 25            ;GO TO SUBROUTINE
                    ;25
X30 = X30+1         ;INCREMENT THE
                    ;LOOP COUNTER
? X30 LE 20 GOTO 12
                    ;EXECUTE LOOP 20
                    ;TIMES
...                 ;CONTINUE PROGRAM
```

10.4.2.2 Nesting ? Commands

You can nest one ? command inside another. For example, suppose you want to break program execution if X1 is less than 100 and greater than -100. You could use:

```
? X1 LT 100 : GOTO 20
? X1 GT -100 : GOTO 20
B
20$
```

However, those four commands can be replaced by just one nested ? command:

```
? X1 LT 100 ? X1 GT -100 B
```

Nesting two ? commands is the same as ANDing the two conditions. The example above only executes the B command if both X1 < 100 and X1 > -100. Nesting of ? commands is limited by the number of entries and the maximum length of a line. VECTOR**STAR** commands are limited to 15 entries (the example above has 9 entries: ? , X1, LT, 100, ?, X1, GT, -100, and B). Since each level of ? command nesting requires 4 entries, you cannot have more than three levels of nesting. Also, a ? command must be less than 80 characters long since it must fit on a single line.

10.4.2.3 TIL Command

The TIL is a single-line command that allows you to specify a condition and a command to be executed repeatedly until that condition is true. The TIL command has the following format:

TIL condition FALSE-command

FALSE-command is repeatedly executed as long as the condition is false. If the condition is true at the beginning of the TIL command, then *FALSE-command* is never executed. In this case, program execution continues to the next step. An example of the TIL command would be to print a line to the operator continuously until the variable PFB is greater than 10000. This statement delays program execution until the condition is true and also refreshes the display while the program waits:

```
TIL PFB GT 10000 P "WAITING FOR PFB >
10000"
```

The TIL command can be used to simply delay your program, because the statement that follows the condition is optional. For example, this statement delays execution, but does not refresh the display:

```
TIL  PFB  GT  10000
```

The TIL can be used to delay program execution.

More examples of the TIL command are:

```
TIL  I1  EQ  ON          ;DELAY EXECUTION
TIL  I1  EQ  ON  P  "PRESS INPUT #1"

TIL  SEG  EQ  0          ;DELAY UNTIL MOTION
                         ;STOPS
TIL  SEG  EQ  0  P  PFB
                         ;PRINT UNTIL MOTION
                         ;STOPS
```

10.4.2.4 IF, ELIF, ELSE, and ENDIF Commands

The IF command, together with ELIF, ELSE, and ENDIF, will allow you to conditionally execute large blocks of commands. These commands are provided because the ? command, which is limited to a single line, does not provide the most efficient means to control blocks of commands. You can use the IF command to write more readable, less error prone programs.

The format of the IF, ELIF, ELSE, and ENDIF commands follows. Note that the conditions have the same format as the conditions for the TIL and ? commands. Note also that *block* can indicate any number of commands:

```
IF  IF-condition
  Block-IF
ELIF  ELIF-condition #1
  ELIF-block #1
ELIF  ELIF-condition #2
  ELIF-block #2
ELSE
  ELSE-block
ENDIF
```

The above example shows two ELIF commands. You can have any number of ELIF commands. The operation of this example IF command is as follows:

If the IF-condition is TRUE,

All commands in the Block-IF are executed.

No other blocks are executed, even if some or all of the other conditions are true.

Program execution continues after the ENDIF command.

Otherwise, if ELIF-condition #1 is TRUE,

All commands in ELIF-block #1 are executed.

No other blocks are executed, even if the conditions that follow are true.

Program execution continues after the ENDIF command.

Otherwise, if ELIF-condition #2 is TRUE,

All commands in ELIF-block #2 are executed.

No other blocks are executed, even if the conditions that follow are true.

Program execution continues after the ENDIF command.

Otherwise,

All commands in ELSE-block are executed.

Program execution continues after the ENDIF command.

Note that only the first block with a true condition is executed. The IF, ELIF, ELSE, and ENDIF commands have several restrictions and options:
Each IF/ELIF/ELSE/ENDIF set...

...must have one and only one IF.
...may have any number of ELIFs.
...need not have any ELIFs.
...may have one ELSE.
...Need not have an ELSE.
...must have one and only one ENDIF.

10.4.2.5 IF vs. ?

You can use ? in place of IF commands. For example, clamping applications make decisions based on the final position of the motor after a move. For our example, assume that the PFB should be between 50 and -50. If

PFB is within range, the program should turn output O1 on and print an appropriate message. If it is out of range, O1 should be turned off and a message should be printed. The table below shows the desired operation:

Table 10.2 Desired Operation of Program Example

PFB RANGE	O1	MESSAGE TO PRINT
PFB > 50	OFF	PFB TOO LARGE
PFB < -50	OFF	PFB TOO SMALL
-50 < PFB < 50	ON	PFB WITHIN RANGE

The IF, ELIF, ELSE, and ENDIF commands implement the desired functions:

```
IF  PFB  GT  50          ;BEGIN BLOCK-IF
 O1  OFF                 ;O1 MEANS "WITHIN
                         ;RANGE"
 P  "PFB EXCEEDED MAXIMUM"
                         ;PRINT ERROR
                         ;MESSAGE
ELIF  PCMD  LT  -50      ;CHECK THE
                         ;NEGATIVE LIMIT
 P  "PFB EXCEEDED MINIMUM"
                         ;PRINT ERROR
                         ;MESSAGE
 O1  OFF                 ;O1 MEANS "WITHIN
                         ;RANGE"
ELSE                     ;IF HERE, THEN
                         ;WITHIN RANGE
 O1  ON                  ;TURN ON O1
 P  "PFB WITHIN RANGE"
                         ;PRINT MESSAGE
ENDIF                    ;END OF BLOCK-IF
```

This example could have been written with ? commands as the following program shows. Notice that the program requires more lines, uses 3 labels, and is harder to read (that is, less intuitive):

```
? PFB  LE  50  GOTO  10$
      ;START OF
;"BLOCK"
 O1  OFF                 ;EXECUTE BLOCK
                         ;IF PFB>50
 P  "PFB EXCEEDED MAXIMUM"
 GOTO  20$               ;DONE—GO TO END
10$
? PFB  GE  -50  GOTO  11$
                         ;EXECUTE BLOCK
                         ;IF PFB<-50
 P  "PFB EXCEEDED MINIMUM"
 O1  OFF
 GOTO  20$               ;DONE—GO TO END
11$                      ;GET HERE IF
                         ;WITHIN RANGE
 O1  ON
 P  "PFB WITHIN RANGE"
20$                      ;END OF "BLOCK"
```

You can choose whether to use ? or the IF command when you are writing your program. You should choose the command that results in the most readable form. For example, if multiple commands are to be executed, the IF command's block structure sets off the commands and avoids the use of a GOTO and a label. On the other hand, if a single instruction is to be executed, the ? may be more readable. Usually, one form results in less program space or faster execution, and this may dictate which to use. However, if space or timing are not critical, use the most readable form.

10.4.2.6 Nesting IF commands

You can nest IF commands. For example, the following program shows two levels of nesting:

```
55$
IF  X1  GT  0
 IF  X2  GT  0
  P  "BOTH X1 AND X2 > 0"
 ELSE
  P  "ONLY X1 GT 0"
 ENDIF
ELSE
 IF  X2  GT  0
  P  "ONLY X2 GT 0"
 ELSE
  P  "NEITHER X1 NOR X2 > 0"
 ENDIF
ENDIF
B
```

You can nest IF commands indefinitely. You should be careful to include all of the ENDIF's to close each level of nested IF. All of the restrictions and options that were listed earlier as applying to IF commands also apply to nested IF's. The indentation shown above is not required but is present to make the program more readable. The VECTOR**STAR** ignores the indentation.

10.4.2.7 IF's with GOTO and GOSUB

You can use the GOSUB command from within a Block-IF, even if you have another Block-IF in that subroutine. In this case, the IF in the subroutine is like a nested IF. However, be careful to return from the subroutine after

you have executed the ENDIF. You should never return from a subroutine from between IF and ENDIF. Finally, you may use a GOTO to jump completely out of an IF-THEN-ELSE control structure. When a GOTO is executed after an IF has been executed, but before an ENDIF has been executed, all ENDIF's are automatically executed. This means that you cannot jump to a label within any IF-THEN-ELSE structure. Note that jumping out of a control structure in such a manner is a poor programming practice and should be avoided. Also, you may not jump to a label within an IF-THEN-ELSE from outside the structure.

You cannot GOTO the middle of an IF/ENDIF set. You should never execute a RET from between an IF and ENDIF.

10.5 USING THE GENERAL PURPOSE INPUTS

General purpose inputs can be used to control the program. From Chapter 8 you may recall that these inputs can be referred to one at a time using variables I1-I16, or collectively IN. If the program must wait for a particular input to be on or off before continuing execution, the TIL command can be used:

```
TIL  I5  EQ  0
```

If this statement is executed from the program, the program will delay execution until I5 is 0.

If the program must wait for many inputs to be on or off, then the TIL command can be expanded. For example, if inputs 1, 4, 5, and 6 must all be on, either of the following TIL instructions can be used:

```
TIL  I1+I4+I5+I6  EQ  4
                  ;THIS USES
                  ;ALGEBRAIC MATH
TIL  I1&I4&I5&I6  EQ  1
                  ;THIS USES
                  ;LOGICAL MATH
                  ; BOTH WORK
```

It is slightly more complicated if the program must wait for some inputs to be on and others off. For example, if inputs 1, 4, and 5 must be on, and input 6 must be off, the following TIL instructions can be used:

```
TIL  I1+I4+I5+(1-I6)  EQ  4
                  ;ALGEBRAIC MATH
TIL  I1&I4&I5&(1-I6)  EQ  1
                  ;LOGICAL MATH
```

Notice the use of (1-I6). This is a logical NOT, because if I6 equals 1, then (1-I6) is 0, and if I6 equals 0, (1-I6) is 1. The logical NOT is useful when checking to see if inputs are off.

If more than a few inputs must be tested, then referencing them one at a time can be cumbersome. As an alternative, IN can be used. This can be demonstrated with the example above. If the program must wait for inputs 1, 4, and 5 to be on and input 6 to be off, logical math can be used to mask the inputs that are not supposed to be tested: inputs 2, 3, and 7-16. A mask is a binary word with a 0 for each input that is not tested and a 1 for each that is. In this example, the mask would be:

Input Number	8	7	6	5	4	3	2	1
Test Input?	N	N	Y	Y	Y	N	N	Y
Binary Mask	0	0	1	1	1	0	0	1

Input Number	16	15	14	13	12	11	10	9
Test Input?	N	N	N	N	N	N	N	N
Binary Mask	0	0	0	0	0	0	0	0

Since the mask must be in hex or decimal, it can be expressed as:

0000000000111001 (BINARY) equals 39 (HEX) or 57 (DECIMAL),

which equals 1+8+16+32 (DECIMAL).

Now that the mask is known, the condition must be determined. The condition is formed much like the mask. In this case, there is a binary 1 for each input that must be on and a binary 0 for each input that is either off or masked:

I/O Number	8	7	6	5	4	3	2	1
I/O Input?	N	N	N	Y	Y	N	N	Y
Binary Mask	0	0	0	1	1	0	0	1

I/O Number	16	15	14	13	12	11	10	9
I/O Input?	N	N	N	N	N	N	N	N
Binary Mask	0	0	0	0	0	0	0	0

Since the condition must be in hex or decimal, it can be expressed as:

0000000000011001 (BINARY) equals 19 (HEX) or 25 (DECIMAL),

which equals 1+8+16 (DECIMAL).

Now the mask and the condition can be used in a TIL instruction in the format:

TIL IN&*mask* EQ *condition*

For our example,

```
TIL  IN&39H  EQ  19H ;THIS USES HEX
                     ;CONSTANTS
```

or

```
TIL  IN&57  EQ  25   ;THIS USES
                     ;DECIMAL.  BOTH
                     ;WORK.
```

This accomplishes the same function as the TIL instruction which refers to inputs one at a time. However, using the IN word allows the function to be done in a less cumbersome manner.

10.6 INTERFACING WITH THE OPERATOR

This section covers interfacing via the serial port (Connector C5). Often, it is necessary to have the VECTOR**STAR** send information to the operator or ask the operator for information. For example, it may be useful to output speed and position, or ask the operator for a new speed command. This is easily accomplished using VECTOR**STAR** serial I/O instructions.

10.6.1 PRINT (P)

The PRINT (P) command prints text and variables to the terminal. Text and variables may be freely intermixed, limited only by the 80-character maximum instruction length. The following command prints the speed on the terminal:

```
P  "SPEED = "  VFB  " RPM"
```

Assuming VFB is 1962, the VECTOR**STAR** will respond with:

SPEED = 1962 RPM

Note that the text must be enclosed by double quotes and that text and/or variables must be separated by at least one blank space. If you want to print only one variable from Motion Link, "P" may be omitted.

10.6.1.1 Printing Decimal Numbers

Variables are normally printed as decimal integers in a field which is 12 characters wide. Formatting can be used to adjust the field width or to print decimal points.

To change the width of the field, follow the variable name with the width enclosed in square brackets ([ ]). Referring to the above example,

```
P  "SPEED = "  VFB[5]  " RPM"
```

will cause the VECTOR**STAR** to print:

SPEED = 1962 RPM

If you try to print a number and do not have enough space in the format for the number, then the VECTOR**STAR** will fill the format width with X's. For example,

```
P  "SPEED = "  VFB[3]  " RPM"
```

will result in:

SPEED = XXX RPM

(again, assuming the speed is 1962 RPM).

10.6.1.2 Printing Decimal Points

You can also use the VECTOR**STAR** to print a decimal point. The VECTOR**STAR** performs calculations with integers because it is much faster than floating point math. However, it is often desirable to convert integers

to floating point numbers, especially when printing out information for the operator. This allows you to make the integer math of the VECTOR**STAR** transparent to the operator. For our example, suppose you would prefer to print out the speed in KRPM (thousands of RPM). You can use print formatting to convert the program units (RPM) to KRPM with the following print command:

```
P "SPEED = " VFB[5.3] " KRPM"
```

Assuming VFB was 1962, this command would produce:

SPEED = 1.962 KRPM.

The .3 which follows the 5 in the format causes the VECTOR**STAR** to insert a decimal point three places from the right of the number. To the operator, this is more convenient, though the programmer still must work in integer units.

You also have the option of printing fewer digits than all those that follow the decimal point. This also can be specified in the format. For example, suppose you only wanted to print one digit after the decimal point. The print command from above would be changed to limit the number of printed digits:

```
P "SPEED = " VFB[5.3.1] " KRPM"
```

This command would produce:

SPEED = 1.9 KRPM

So the general order for decimal format is:

[OVERALL WIDTH.DECIMAL POSITION.PRINTABLE DIGITS]

For the example above ([5.3.1], the overall width was 5, the decimal position was 3, and the number of printable digits after the decimal was 1. You can leave off any of these three specifications. The overall width defaults to 12, the decimal position to zero, and the printable digits to the value of the decimal position.

10.6.1.3 Printing Hex Numbers

To print a variable in hexadecimal, follow the variable name with an H enclosed in square brackets ([H]). The variable will be printed in a field 9 characters wide, including an appended H, indicating hex. The default field width of 9 can be changed by following the H with the desired field width. For example:

```
X1 = 255
P "X1 = " X1[H]
P "X1 = " X1[H3]
```

will cause the VECTOR**STAR** to print:

```
X1 =      FFH
X1 = FFH
```

Two's-complement notation is used when printing in hex. This means that printing negative hex values requires the full field width of 9 characters. When printing in hex format, the field must be wide enough to include the appended H.

10.6.1.4 Printing Binary Numbers

To print a variable in binary format, follow the variable name with a B enclosed in square brackets ([B]). The variable will be printed in a field 33 characters wide, including an appended B, indicating binary. All of the leading zeros will be printed. The default field width of 33 can be reduced by following the B with the desired field width. For example:

```
X2 = 127
P "X2 = " X2[B]
P "X2 = " X2[B10]
```

will cause the VECTOR**STAR** to print:

```
X2 = 000000000000000000000001111111B
X2 = 001111111B
```

10.6.1.5 Printing Switches

Formatted printing can also be used to display switches (any variable with a value of 0 or 1) either as Y or N or as on or off. This allows you to communicate with the operator better than just printing 0 or 1. The switch format (on or off) is printed with a bracketed S ([S]) following the variable:

```
XS1 = 1
P "USER SWITCH #1 IS " XS1[S]
```

These commands would result in:

```
USER SWITCH #1 IS  ON
```

Similarly,

```
XS1 = 0
P  "USER SWITCH #1 IS "  XS1[S]
```

results in:

```
USER SWITCH #1 IS OFF
```

In addition, you can print a switch as Y or N if you follow the switch with a bracketed Y ([Y]). For example,

```
P  XS1[Y]
```

will print either Y or N depending on whether XS1 is 1 or 0, respectively. This format is useful with the input command which we will discuss later. The input command allows the operator to respond with Y or N and stores 1 or 0 in a VECTOR**STAR** variable. This print format allows you to print the previous answer on the screen the way it was entered.

10.6.1.6 Printing Expressions

The P instruction is not restricted to printing only variables. In general, any numeric expression can be formatted and printed. All the following examples are valid:

```
P  "MINUS 1 IN HEX IS "  -1[H]
P  X1+X3  " IS THE RESULT OF ADDING X1
AND X3"
P  "SENSE OF DIRECTION IS "  DIR*2-1[2]
P  "DISTANCE TO GO IS "  PFNL-PFB[.3]
"INCHES"
P  "HIGH BYTE OF IN IS "  (IN&0F0H)/
10H[H3]
```

10.6.1.7 Printing ASCII Characters

The VECTOR**STAR** will also convert numbers to ASCII format before printing. You can do this by following the variable or expression with a bracketed C ([C]). This will cause the VECTOR**STAR** to print out the character for which the number is an ASCII code. For example,

```
X6 = 65
P  "THE NUMBER " X6[2] " IS THE ASCII
CODE FOR "  X6[C]
```

will result in:

```
THE NUMBER 65 IS THE ASCII CODE FOR A
```

If the number is greater than 127 (that is, the eighth bit is set), the VECTOR**STAR** removes the eighth bit before transmitting the character. For example:

```
P  65[C]  " IS THE SAME AS "  128+65[C]
```

since the VECTOR**STAR** removes the eighth bit of the expression on the right, which has the end effect of reducing the number by 128. If the number is larger than 255, the VECTOR**STAR** divides the variable or expression into four bytes and prints them out separately. For example:

```
X2 = 256*256*256*65+256*256*65+256*65+65
P  X2[C]
```

prints:

```
AAAA
```

since the number stored in X2 is equivalent to 4 bytes of 65.

The default field width of the character format is 4, and you can change the field width by following the C with the desired format.

10.6.1.8 Printing Control Characters

The VECTOR**STAR** uses the standard ASCII character set as shown in Appendix E. There are unprintable characters, such as the bell (ASCII 7) and carriage return (ASCII 0DH). These characters have an effect on the terminal but do not print anything on the screen. Unprintable characters range from ASCII 1 to 1FH. The VECTOR**STAR** cannot print ASCII 0.

As Appendix E shows, each unprintable character can be produced with a control sequence. For example, most terminals will sound a bell when you press <Control>G (hold down the control key while pressing the G key). As Appendix E shows, <Control>G produces 07 or the ASCII bell. You can use the VECTOR**STAR** to produce unprintable characters by preceding the appropriate character with the carat (^) to signify an unprintable character. For example, the following VECTOR**STAR** command will sound the bell on your terminal:

```
P  "^G"
```

You can also use the character format to print control characters. For example:

```
P  07[C]
```

also sounds the bell. The character format allows you to print variables as ASCII codes. However, the easiest way to print control characters is normally with the carat (^). One reason for this is that control characters can be within text strings. For example:

```
P "BELL = <CONTROL>G. ^G SOUNDS A BELL"
```

If you use the carat to specify an invalid control character, such as ^1, the VECTOR**STAR** will print the carat and the 1 ("^1"). Only ^A to ^Z, ^[, ^/, ^], ^^ and ^_ are allowed.

10.6.1.9 Cursor Addressing

Many displays allow you to address the cursor. For example, the DEP-01 from Kollmorgen is an 80-character display that allows you to address any location from 0 (leftmost top line) to 79 (rightmost bottom line). First, send ASCII 27 ("^[") followed by the address of ASCII 0 ("^@") through ASCII 79 ("O"). For example, you can address the rightmost space of line one (space #39) with the control character sequence ^['. The ^[specifies cursor addressing and ' (ASCII 39) specifies space #40.

One problem with cursor addressing is that the VECTOR**STAR** cannot transmit ASCII 0 (^@). This is a common limitation for terminals. If you want to address space #0, you must first address space #1, then transmit a backspace (ASCII 8 or "^H"). For example, if the following line is executed from the user program while the VECTOR**STAR** serial port is connected to the DEP-01, "X" will be printed on space #0.

```
P "^[^A^HX MARKS THE FIRST SPACE"
```

10.6.1.10 Printing VECTORSTAR Status (PS)

The PRINT STATUS (PS) command is like the P command except that it appends the VECTOR**STAR** status to the end of the printed line. There are five different status words that can be printed with the PS command. Each is listed with its meaning in Table 10.3. You can use all formats and combinations with PS that you did with P. These results are identical except that the VECTOR**STAR** status is appended onto the line.

Table 10.3 Printing VECTORSTAR Status

Status	Explanation
OFF	VECTOR**STAR** IS OFF
READY	VECTOR**STAR** is ready, but REMOTE if OFF
ACTIVE	VECTOR**STAR** is active, but no motion.
FAULT	VECTOR**STAR** has a fault condition.
JOG	VECTOR**STAR** is jogging
PROFILE	VECTOR**STAR** is executing profile.
GEAR	VECTOR**STAR** is in gear mode.

10.6.2 REFRESH (R & RS) Commands

The REFRESH commands, R and RS, are identical to P and PS, except that R and RS send only a carriage return. The P and PS commands print lines that end with linefeed and carriage return pairs. R and RS commands display lines that can be overwritten.

The following example demonstrates how the REFRESH commands work. Type in this example from the Editor:

```
7$
RS "VELOCITY FEEDBACK=" VFB
GOTO 7
```

Now exit the Editor and type:

```
RUN 7
```

Rotate the motor shaft by hand so that the velocity feedback changes. Press the escape key and enter the Break command to break program execution. Notice that the velocity is continuously updated, but the line appears to be stationary. A similar program with the P or PS commands would cause the lines to scroll to the top of the screen.

10.6.3 INPUT

So far, printing information to the operator has been discussed. This section will discuss how to prompt the operator for information using the INPUT command. The INPUT command causes the VECTOR**STAR** to print a message to the terminal and wait for a response

from the operator. The input information can be stored in any programmable variable. This allows the operator to change or enter information without making any changes to the program itself. You can only execute the INPUT command from the user program.

Type in the following example INPUT instruction:

```
INPUT  "ENTER NEW SPEED : "  X2
```

This causes the VECTOR**STAR** to print :

```
ENTER NEW SPEED :
```

Type the new speed into the terminal. After you are prompted, enter a number and press the enter key. The number you enter is stored in the variable X2. If you press the enter key without entering a number, the variable X2 is left unchanged. Use the Print command to display the new value of X2:

```
P  X2
```

10.6.3.1 INPUT Limits

You can also specify an upper and lower limit for the operator entry. If the above INPUT instruction were written as:

```
INPUT  "ENTER NEW SPEED : "  X2  10  100
```

the VECTOR**STAR** would force the operator to input a value between the specified low limit (10) and high limit (100). If the input is invalid or outside the range, an error message is sent and the operator is prompted again.

The limits can be constants, as shown above, as well as any valid numerical expression. If the limits are outside the variable's normal range, they are ignored. If they are not specified at all, the variable's normal range is used as the limit. For example, the limits on ACC are 0 and AMAX. Type in this command:

```
X1=ACC                ;STORE ACC
INPUT  "ENTER NEW ACC : "  ACC  -1000
1000
```

The VECTOR**STAR** knows that the lower limit on ACC is 0 so that no negative numbers will be accepted. If AMAX is less than 1000, AMAX will be the upper limit. Otherwise, 1000 will be the upper limit. If you specify limits that are outside the variable's program limits, the VECTOR**STAR** uses the program limits. Appendix F lists all variables and their program limits.

10.6.3.2 INPUT and Decimal Point

You can use the INPUT to prompt the operator for values that include a decimal point. You must specify the number of characters after the decimal point. This is the only way you can enter numbers having a fractional part into the VECTOR**STAR**. For example, suppose your user position units are mils (0.001 inches). You can prompt the operator for any position in inches with the INPUT. The following example stores the results of the INPUT command in X1. Enter this short program in your VECTOR**STAR**; then type RUN 44:

```
44$
INPUT  "ENTER NEW POSITION: "  X1[3]
P  "NEW POSITION = "  X1[.3]
P  "ACTUALLY, X1= "  X1
B
```

Notice the bracketed 3 following X1 in the INPUT command. This causes the operator input to be multiplied by 1000 (10^3) before it is stored in X1. The print statements that follow display X1 in inches (as the operator would prefer to see it), then in mils (as the VECTOR**STAR** motion commands process it).

10.6.4 SERIAL Switch

You can use the SERIAL switch to make sure that the serial port is not busy before you execute a command. If SERIAL is on, the serial port is ready. For example, suppose you do not want to execute an INPUT command if the serial port is busy. It might be busy from a print command, or from a previously executed input command. In that case, use these commands:

```
? SERIAL EQ ON INPUT "ENTER SPEED" X1
```

10.7 IDLING COMMANDS

There are four idling commands: HOLD (H), DWELL (D), WAIT (W), and INPUT. This section discusses the first three. The INPUT command was discussed above. HOLD, DWELL, and WAIT cause the user program to wait for an event before executing the next command. HOLD waits for switches, DWELL waits for a timer, and WAIT waits for a motion segment.

10.7.1 HOLD (H)

The HOLD command waits for a switch to be either on or off. You specify the HOLD command with the switch and the desired state. For example,

```
H  I1  ON          ;HOLD UNTIL INPUT
                   ;I1 IS ON
H  O2  OFF         ;HOLD UNTIL OUTPUT
                   ;O2 IS OFF
H  TRIP1  ON       ;HOLD UNTIL PFB >
                   ;PTRIP1
```

Use the VECTOR**STAR** to enter the following program:

```
29$
P  "TURN I1 ON"
H I1 ON
P  "I1 IS NOW ON"
B
```

Exit the Editor, turn input I1 off, and observe the action of the HOLD command by typing:

```
RUN 29
```

You can Hold for any switch except REMOTE and user switches (XS11-XS50). User switches XS1-XS10 are allowed with the HOLD command.

10.7.2 DWELL (D)

Sometimes it is desirable to delay execution for a specified amount of time. The DWELL (D) command; is the easiest way to do this. The delay is specified in milliseconds. For example:

```
D  1000        ;DWELL 1000 MILLISECONDS
```

delays execution for 1000 milliseconds or 1 second. The DWELL command can be demonstrated by typing in the following simple program:

```
6$
P  "BEGIN 5 SECOND DWELL"
D  5000
P  "END   5 SECOND DWELL"
B
```

Now exit the Editor and type:

```
RUN  6
```

The result should be:

```
BEGIN   5 SECOND DWELL
END     5 SECOND DWELL
```

with 5 seconds between lines being printed. Dwells can be up to 2,147,483,647 milliseconds or about 25 days.

10.7.3 WAIT (W)

When using MOVE commands, it is often necessary to synchronize the execution of your program to motion. The WAIT (W) command can be used to wait for the specified motion segment. Examples of the WAIT command are:

```
W  0           ;WAIT FOR MOTION TO STOP
W  1           ;WAIT FOR MOTION
               ;COMMAND TO BEGIN
W  14          ;WAIT FOR SEGMENT 14
               ;(MACRO MOVE)
```

These commands are similar; *W 0* delays program execution until the last motion command entered has stopped. *W 1* delays program execution until the last motion command entered has started. *W 14* waits for segment 14 of the last motion command to begin.

In the example below, the WAIT command is used to delay the calculations of the third move until the second move has begun. The use of *W 1* here allows the third move to be calculated while the second is being executed. **Do not type in the following example**—it is meant to run as a part of the user program.

```
MI  10000  100     ;BEGIN THE FIRST
                   ;MOVE
MI  10000  200     ;CALCULATE THE
                   ;SECOND MOVE
                   ;WHILE THE FIRST IS
                   ;IN PROGRESS
W  1               ;DELAY PROGRAM
                   ;EXECUTION UNTIL
                   ;THE SECOND
                   ;MOVE HAS
                   ;STARTED
MI  10000  300     ;CALCULATE THE
                   ;THIRD MOVE AND
                   ;PREPARE IT FOR
                   ;EXECUTION
```

The WAIT (W) command and synchronization will be discussed in more detail later in this chapter.

10.8 MULTI-TASKING

Multi-tasking is an important feature of the VECTOR**STAR**. Multi-tasking allows you to write separate *tasks* that run *concurrently*, which means more than one task executes at the same time. For example, you can write a program with two separate tasks: one to ask the operator questions and another to command motion. These two tasks can run independently so that while the operator is answering questions, the motion continues.

Each task has a priority level. The VECTOR**STAR** has 6 different task levels as detailed in Table 10.4. High priority means that if two tasks both need to run at the same time, then the commands from the task with highest priority will execute first. For example, Alarm A has the highest priority. If Alarm A and Alarm B are "fired" at the same time, Alarm A will run until it is complete; then Alarm B will run until it is complete.

10.8.1 Multi-Tasking and Autobauding

If you set the VECTOR**STAR** to autobaud, multi-tasking will not be enabled until communications have been established. This means that the VECTOR**STAR** will not operate if a terminal or computer is not present. Therefore, you normally will want to disable autobauding by turning ABAUD off.

Turn ABAUD off if you plan to use multi-tasking. The VECTORSTAR will remember that ABAUD is off through power-up.

10.8.2 MULTI

If you want to disable Alarm C, the variable input routine and background, type:

```
MULTI OFF
```

For example, if you have a time critical section of code, you may turn MULTI off at the beginning of the section and then back on at the end of the section.

10.8.3 END Command

Tasks are normally terminated with the END command. END signifies the end of the task, whereas Break (B) implies that all tasks stop executing. For example, if you end an alarm with the Break command, the entire program stops running and the VECTOR**STAR** returns to the Interactive mode. However, if you end an alarm with the END command, the alarm stops, but the other tasks continue running.

10.8.4 Enabling and Disabling Multi-tasking

Multi-tasking is always enabled when a program is running. For example, if you have a program that starts at 55$ and has 2 alarms, then the alarms will be active if you type:

```
RUN 55
```

If your program ends with a Break command, then the program will stop executing and multi-tasking will be disabled; that is, the VECTOR**STAR** will return to the Interactive mode. If your program ends with an END command, then only the task level that executed the END will stop executing; other tasks will continue executing. If there are no other tasks that are executing, then the VECTOR**STAR** does not return to the Interactive mode but instead becomes dormant. In this case, multi-tasking remains enabled. For example, alarms will continue to be serviced.

If you want to enable multitasking without running a particular program, type:

```
RUN
```

without entering a label.

Table 10.5 shows how to turn multi-tasking on and off.

Table 10.4 Multi-Tasking Overview

Task Level	Task Name	Task Labels	How to Start Task	Typical Uses of Task
1 (Highest Priority)	ALARM A	A$	Hardware or Software Switch	Monitor Inputs
2	ALARM B	B$	Hardware or Software Switch	
3	ALARM C	C$	Hardware or Software Switch	
4	VARIABLE INPUT	VARIABLE$	"ATTN" from DEP-01 or ^V from a PC or a Terminal	Prompt Operator for Input
5	POWER-UP PROGRAM	POWER-UP$	Power-up VECTOR**STAR** and Establish Communications	Initialize VECTOR**STAR** for Application
	AUTO PROGRAM	AUTO$	Manual Switch Off and Positive Transition Of Cycle Input	Run One Cycle of Auto Program
	MANUAL PROGRAM	MANUAL$	Manual Switch On	Run Manual Program Continuously
	GENERAL PURPOSE PROGRAM	0$ - 500$	Run <LABEL>	General Purpose Programs
	USER ERROR HANDLER	ERROR$	Any Error That Breaks Execution	Gracefully Exit on Error Condition
6 (Lowest Priority)	BACKGROUND PRINT AND MONITOR	BACKGROUND$	All Other Tasks Idle	Print Messages to the Screen

Table 10.5 How to Enable and Disable Multi-Tasking

How to Enable Multi-Tasking	
1.	Run any label (Type "RUN <label>").
2.	Run multitasking (Type "RUN").
3.	Include a POWER-UP$ label and power-up.
How to Disable Multi-Tasking	
1.	Execute a Break from your program.
2.	Enter a Break from the Monitor mode.
3.	Cause an error that breaks execution.

10.8.5 Idling

Idling is a necessary part of multi-tasking. So far in our discussion, higher priority tasks run until they are complete. Actually, commands from the highest priority task that is not idle execute. For example, if an alarm cannot run because it is waiting for some condition (such as waiting for motion to stop), it is *idle*. If a task is running, and it becomes idle, then a lower priority task can run until the higher priority task is no longer idle. A task can be idled with pre-execution idling and post-execution idling.

10.8.5.1 Pre-Execution Idle

A task can be idled by waiting for a condition before executing a command. This is called a "pre-execution idle" because the task is idled before executing the command that causes the idle. There are two conditions that can cause a pre-execution idle. A task about to execute a motion command (MI, MA, or MCGO) will be idled if the motion buffer is full. Also, a task about to execute a printing command (P, PS, R, RS, or INPUT) will be idled until the previous printing command is finished.

For example, the VECTOR**STAR** can store up to two MI or MA commands. This is called *buffering* in Chapter 8. This means that if you wrote a task with three MI commands in a row, then the third MI command could not be executed until the first move was complete. So that task would be idled until the first move finished. If there was another, lower-priority task, it would execute until the first move finished. When the first move finished, the first task would no longer be idled, and thus would proceed.

Consider the following program, which has two tasks: a routine starting at 1$ (task level 5) and a background task starting at *BACKGROUND$* (task level 6). The background task is the lowest priority task and will only execute when the general purpose task is idle. In the following example, the task is idle between the second and third motion command. Use the VECTOR**STAR** Editor to enter this program:

```
;TASK LEVEL 5

1$                  ;MAIN PROGRAM
EN
MI  10000  10       ;FIRST MOVE
P  "FIRST MOVE PROCESSED"
MI  10000  10       ;SECOND MOVE
P  "SECOND MOVE PROCESSED"
MI  10000  10       ;THIRD MOVE
P  "THIRD MOVE PROCESSED"
B
.....

;TASK LEVEL 6

BACKGROUND$
P  "UPPER TASK IDLED"
D  250              ;DWELL 0.25 SEC.
END
```

Apply DC bus power to your VECTOR**STAR** and type:

```
RUN 1
```

The result should be:

FIRST MOVE PROCESSED
SECOND MOVE PROCESSED
UPPER TASK IDLED
UPPER TASK IDLED
. . .
UPPER TASK IDLED
UPPER TASK IDLED
THIRD MOVE PROCESSED

The first and second moves are processed immediately. Then task level 5 is idled while the first move finishes. While task level 5 is idle, the background task executes over and over, printing the simple message on the screen.

10.8.5.2 Post-Execution Idle

A task also can be idled by waiting for a condition after executing a command. This is called a "post-execution idle" because the task is idled after executing the command that causes the idle. Commands that cause post-execution idling are called *idling commands*. There are four idling commands:

WAIT (W), DWELL (D), HOLD (H), INPUT(I).
For example, you can modify the above program to make one move, then run the background routine until motion has stopped. Use the VECTOR**STAR** Editor to enter the following program.

```
;TASK LEVEL 5

1$                      ;MAIN PROGRAM
EN
MI  10000  10           ;START MOVE
P  "MOVE PROCESSED"
W  0                    ;WAIT FOR MOVE
P  "ALL MOTION STOPPED"
B

.....................

;TASK LEVEL 6

BACKGROUND$
P  "UPPER TASK IDLED"
D  250                  ;DWELL 0.25 SEC.
END
```

Apply DC bus power to your VECTOR**STAR** and type:

```
RUN 1
```

The result should be:

MOVE PROCESSED
UPPER TASK IDLED
UPPER TASK IDLED
. . .
UPPER TASK IDLED
UPPER TASK IDLED
ALL MOTION STOPPED

Note that task level 5 immediately processes the move and then is idled until motion stops. While task 5 is idled, the lower level, background task executes continuously.

10.8.5.3 Avoiding Idling

You can avoid idling the VECTOR**STAR** by using the TIL command in place of Dwell, Wait, or Hold. For example,

```
TIL  SEG  EQ  0
```

is the same as:

```
W  0
```

except the TIL command locks out lower priority tasks since it is not an idling command. The Wait command allows lower level tasks to execute since it is an idling command.

10.8.6 Alarms (Task Levels 1-3)

Alarms are the highest priority tasks. There are three alarms: A, B, and C. A is the highest priority and C is the lowest. Normally, alarms are used to monitor hardware inputs, but they can monitor any user switches (XS1 - XS50) and MANUAL. Using an alarm relieves you of having to write your program so that it checks switches. After you define an alarm, the VECTOR**STAR** will watch the switch and automatically execute the code that you specify, should the alarm "fire."

Alarms are specified on one line, along with the switch that triggers the alarm and the transition. For example, the A alarm can be defined to fire when input I1 transitions from off to on with this command:

```
A$  I1  ON
```

You can follow the alarm definition with the code that you want to execute when the alarm fires. For example, if I1 turned on, it might indicate an error condition. In this case you might disable the VECTOR**STAR**, turn off all outputs, and break execution. The following program would accomplish this using the A alarm.

```
A$  I1  ON          ;DEFINE THE ALARM
DIS                 ;DISABLE THE
                    ;VECTORSTAR
OUT = 0             ;TURN OFF ALL
                    ;OUTPUTS
B                   ;BREAK EXECUTION
```

10.8.6.1 Restrictions of Alarms

Alarms have many restrictions. 1) You cannot execute GOTO, GOSUB, or RET commands from an alarm. 2) You cannot execute a label. 3) You cannot use the REMOTE switch to fire an alarm. 4) Alarms must be self-contained programs—they cannot "mix" with your program. 5) They must be terminated with an END, KILL (K), or BREAK (B) command. 6) Also, if all three alarms are present, the execution time of your program increases by about 3%. Most other commands are allowed for alarms, including motion commands and Block-IFs.

10.8.6.2 Printing with Alarms

You must be careful when executing print commands from alarms. If you need to print from an alarm task, always print after the critical commands have been executed. This is necessary because the input command from a lower task will stop any task, even a higher priority task, from printing. The input command stops all printing until the operator responds with a new value. For example, write your program like this:

```
B$ HOME ON          ;FIRE ALARM WITH
                    ;HOME
O = 0               ;TURN OFF
                    ;OUTPUTS
DIS                 ;DISABLE DRIVE
P "MESSAGE"         ;NOW PRINT A
                    ;MESSAGE
B
```

Do not print before you turn outputs off or disable the VECTOR**STAR**. Otherwise, an INPUT command from another task may idle the alarm indefinitely.

10.8.7 Variable Input (Task Level 4)

The variable input task is the next highest priority. Normally, the variable input task is used to prompt the operator for input, while still allowing the main section of the program to continue. For example, the operator could be entering a new distance while the main program continues executing the program using the old distance. The variable input task is similar to an alarm, except that it is fired upon receiving a special character from the terminal or computer, which is ^V (control-V), or ASCII 16H. The "ATTN" button on the Kollmorgen DEP-01 Data Entry Panel also transmits a ^V to fire the variable input task.

The variable input task begins with *VARIABLE$*. You can then follow that label with various statements, usually printing and input commands. For example, enter the following program:

```
;TASK LEVEL 4

VARIABLE$
P  "X1 IS"  X1
INPUT  "INPUT NEW VALUE OF X2" X2
P  "X1 IS NOW "  X1
B                       ;END EXECUTION

.....................

;TASK LEVEL 5

10$
X1 = 0
11$
X1 = X1+1
GOTO 11
```

Now you can enable multitasking by typing:

```
RUN  10
```

This program resets X1, then begins to count up. Now enter ^V from your terminal or ATTN from your DEP-01. The VECTOR**STAR** should print the value of X1 which has been continuously incrementing since you typed RUN 10. Next, enter a new value for X2 and notice that the program prints out a new value for X1, which is larger than the value it printed at the beginning of the variable input task. This is because the variable input task was idle while you were entering the new value. Since the higher priority task is idle, the lower priority (11$) will run and continuously increment X1.

10.8.7.1 Using Variable Input with Profiles

You can use the variable input routine while the VECTOR**STAR** is executing motion profiles. However, you must be careful if you are changing parameters of motion. Specifically, if you are changing two or more parameters which you want to take effect at the same time, you must write your program to store those values away. For example, suppose you are using the variable input routine to prompt for speed and distance. You might use a program like this:

```
;TASK LEVEL 4

VARIABLE$
INPUT  "INPUT NEW DISTANCE"  X1
INPUT  "INPUT NEW SPEED"  X2
END                    ;END VARIABLE$

.....

;TASK LEVEL 5

20$
MI  X1  X2
GOTO 20
```

If you type:

```
RUN  20
```

this program will continuously move the motor X1 distance at X2 speed, even after you press ^V to start the variable input routine. However, after you have entered a new value for X1, the variable input routine will be idled, waiting for you to enter X2. In this case, the next MI command will be executed with the new X1 and the old X2. You can correct this problem by temporarily storing the input values in user variables and loading them all together. For example, the above program can be modified as follows:

```
;TASK LEVEL 4

VARIABLE$
INPUT  "INPUT NEW DISTANCE"  X11
INPUT  "INPUT NEW SPEED"  X12
X1 = X11               ;LOAD X1 AND X2
     ;WITH
X2 = X12               ;INPUT VALUES
END                    ;END VARIABLE$

.....

;TASK LEVEL 5

20$
MI  X1  X2
GOTO 20
```

Temporarily storing the input values in X11 and X12 guarantees that the MI command will execute with either all new or all old values. Since there are no idling commands between the commands that load X1 and X2, there is no possibility for task level 5 to run until X1 and X2 are both loaded or neither is loaded.

In addition, if the variable input routine changes variables used in different lines of task level 5, you probably should turn MULTI off at the beginning of the block of lines and back on at the end. This prevents the variable input routine from reloading the variables in the middle of a block of lines.

10.8.7.2 Restrictions of Variable Input

Like alarms, variable input has many restrictions. 1) You cannot execute GOTO, GOSUB, or RET commands from the variable input task. 2) You cannot execute a label. 3) The variable input must be self-contained—it cannot "mix" with other tasks. It must be terminated with an END, KILL (K), or BREAK (B) command. Again, most other commands are allowed for the variable input task, including motion commands and Block-IFs. If the variable input task is present, the execution time of your program increases by about 1%.

10.8.8 Main Program Level (Task Level 5)

Most of the time, your program will run at task level 5. All the program examples given earlier in this chapter executed at task level 5. Notice the discussion on multi-tasking that all general purpose labels (0$ - 500$) and many dedicated labels (POWER-UP$, AUTO$, MANUAL$, and ERROR$) share task level 5. The routines that follow these labels share one task level and cannot run concurrently. For example, you cannot run AUTO$ and MANUAL$ concurrently. In other words, only one task-level-5 routine can run at a time.

Alarms and the variable input task are higher priority than task level 5. For example, if an alarm fires while your program is running a task that begins at a general purpose label (task level 5), task level 5 will be suspended until the alarm is complete. The background program (BACKGROUND$) runs at the lowest level. Generally, alarms respond to conditions that are more urgent than most other sections of the program. Similarly, background is for tasks that are not critical, such as printing. Multi-tasking controls which task runs by executing commands from the highest priority task that is not idle.

The rest of this section will discuss the dedicated labels in task level 5: POWER-UP$, ERROR$, AUTO$, and MANUAL$.

10.8.8.1 Power-Up Routine (POWER-UP$)

On power-up, the VECTOR**STAR** checks your program to see if you entered POWER-UP$. If you did, the power-up routine is executed. For example, enter the following program:

```
POWER-UP$
X1 = X1+1              ;SAMPLE COMMAND
B
```

Now power-down your VECTOR**STAR** for a few seconds and power-up again. After establishing communications, the VECTOR**STAR** should display the sign-on message followed by:

```
EXECUTING POWER-UP LABEL
—>
```

indicating that the power-up routine was executed.


NOTE

The power-up label is run after the autobaud.

If you want your program to start automatically on power-up, begin it with POWER-UP$. If POWER-UP$ is not found in the program, the VECTOR**STAR** powers-up in the Interactive mode. If the VECTOR**STAR** is set to autobaud, it will not execute the power-up label until communications have been established.

If you want to leave multi-tasking active after your power-up routine is done, end the power-up routine with the END command instead of the Break command. If your routine ends with the END command, then multi-tasking will be enabled, and the Alarms, Background, and other multi-tasking functions will be working. If you want to return to the Interactive mode after power-up, then end the power-up routine with the Break command.

10.8.8.2 Error Handler (ERROR$)

When a serious error occurs, the VECTOR**STAR** breaks execution of your program and checks your program to see if you entered ERROR$. If you did, the error handler (the routine that follows the ERROR$) is executed. All multi-tasking is suspended, including alarms, when the error handler is being executed.

10.8.8.3 Auto Routine (AUTO$)

If you want to start a program from an external switch, you should use the auto routine. You can use the auto routine to interface to simple operator panels or to programmable logic controllers (PLCs).

CYCLE (Connector C7, Pin 13) is a hardware input that, under the proper conditions, will cause the VECTOR**STAR** to begin executing one cycle of the auto program. The AUTO program begins at AUTO$. CYCLE READY is a hardware output that indicates the VECTOR**STAR** is ready to run another cycle of the AUTO program.

The following conditions must be met for the VECTOR**STAR** to execute the AUTO program. When these conditions are met, the CYCLE READY output (Connector C7, Pin 23) will turn on.

1. Multitasking must be enabled.
2. AUTO$ must be present in the user program.
3. No routines can be executing at task level 5.
4. The MANUAL input must be off.
5. The CYCLE input must be low.

If these conditions are met, the CYCLE READY output will turn on. Then, when CYCLE turns on, the VECTOR**STAR** will begin executing the user program at AUTO$, and CYCLE READY output will turn off.

10.8.8.4 Manual Program (MANUAL $)

The following conditions must be met for the VECTOR**STAR** to execute the MANUAL program. When these conditions are met, the VECTOR**STAR** will begin executing label MANUAL$.

1. Multitasking must be enabled.
2. MANUAL$ must be present in the user program.
3. No routines can be executing at task level 5.
4. The MANUAL input must be off.

If these conditions are met, the VECTOR**STAR** will execute the user program at MANUAL$. You may have noticed that AUTO and MANUAL are very similar. The important difference is that while the AUTO program begins when CYCLE START turns on, the MANUAL program runs continuously.

10.8.8.5 Typical AUTO/MANUAL Programs

Figure 10.2 shows typical AUTO and MANUAL programs. This flowchart shows the effects of the MANUAL and CYCLE switches. The sample AUTO program causes the motor to rotate one revolution each

time the CYCLE switch transitions from off to on. The sample MANUAL program is written so that I1 and I2 are JOG+ and JOG- switches. So when the MANUAL switch is on, the VECTOR**STAR** monitors the jog buttons; when MANUAL is off, the CYCLE button causes the motor to rotate one revolution. Note that both the AUTO and MANUAL programs end with the END command; this is the normal way to conclude these programs.

10.8.9 Background (Task Level 6)

The background task is the lowest priority. Normally, the background task is used for non-critical tasks such as refreshing the display and checking low priority inputs. The background task runs continuously, as long as no other task is active.

The background task begins with *BACKGROUND$*. You can then follow that label with various statements, usually printing commands. For example, enter the following program:

```
BACKGROUND$
P  "EXECUTING BACKGROUND"
D  500
        ;DWELL
END
```

Now you can enable multitasking by typing:

```
RUN
```

Notice that you did not need to specify a label. If you type RUN without a label, you will enable multi-tasking without executing a specific label. When you are done with this example, press ^X (control X) to break the program and return to the Interactive mode.

10.8.9.1 Restrictions of Background

Like alarms, background has many restrictions.

1. You cannot execute GOTO, GOSUB, or RET commands from background.
2. You cannot execute a label.
3. The background task must be self-contained; it cannot "mix" with other tasks. It must be terminated with an END, Kill (K), or Break (B) command.

Again, most other commands are allowed for the background task, including Block-IFs. If the background task is present, the execution time of your program increases by about 1%.

10.9 UNITS

The VECTOR**STAR** provides user units so that both you and the machine operator can work in units that are convenient. The VECTOR**STAR** allows you to define the units of acceleration, current, velocity, and position for your machine. Also, if your VECTOR**STAR** has an external input, you can define the units of external position and external velocity.

10.9.1 User Units

The VECTOR**STAR** uses internal units, called *VECTOR**STAR**-basic units*, which are very inconvenient to use. For example, velocity is in (1/65.536)*counts/second. User unit constants scale the VECTOR**STAR**-basic units. For example, if you type:

```
VOSPD = 1000
```

the 1000 is multiplied by VNUM/VDEN before it is stored in the VECTOR**STAR** memory. Your VECTOR**STAR** is shipped with VNUM and VDEN set so that the user velocity units are RPM. However, with a simple, step-by-step procedure, you can redefine the units as inches/minutes, degrees/second, or any other units that are convenient for your machine. The following table shows some common user units.

Table 10.6 Common User Units

Current	Percent Amps	INUM=4095 INUM=4095	IDEN=100 IDEN=FULL AMPS
Position	Counts	PNUM=1	PDEN=1
Velocity (12-Bit)	RPM rad/sec	VNUM=44739 VNUM=42723	VDEN=10 VDEN=1
Accel (12-Bit)	RPM/sec rad/(secΔsec)	ANUM=4474 ANUM=4272	ADEN=1000 ADEN=100

10.9.1.1 Current Units

The VECTOR**STAR** commands current with a 12-bit digital-to-analog converter (DAC). The VECTOR**STAR**-basic current unit is 1/4095th of full-scale current. Full-scale current refers to the peak rating of your VECTOR**STAR**, not the continuous rating. For example, the peak rating of a 6 Amp VECTOR**STAR** is 12 Amps.

The conversion constants that determine user current units are INUM, current units numerator, and IDEN, current units denominator:

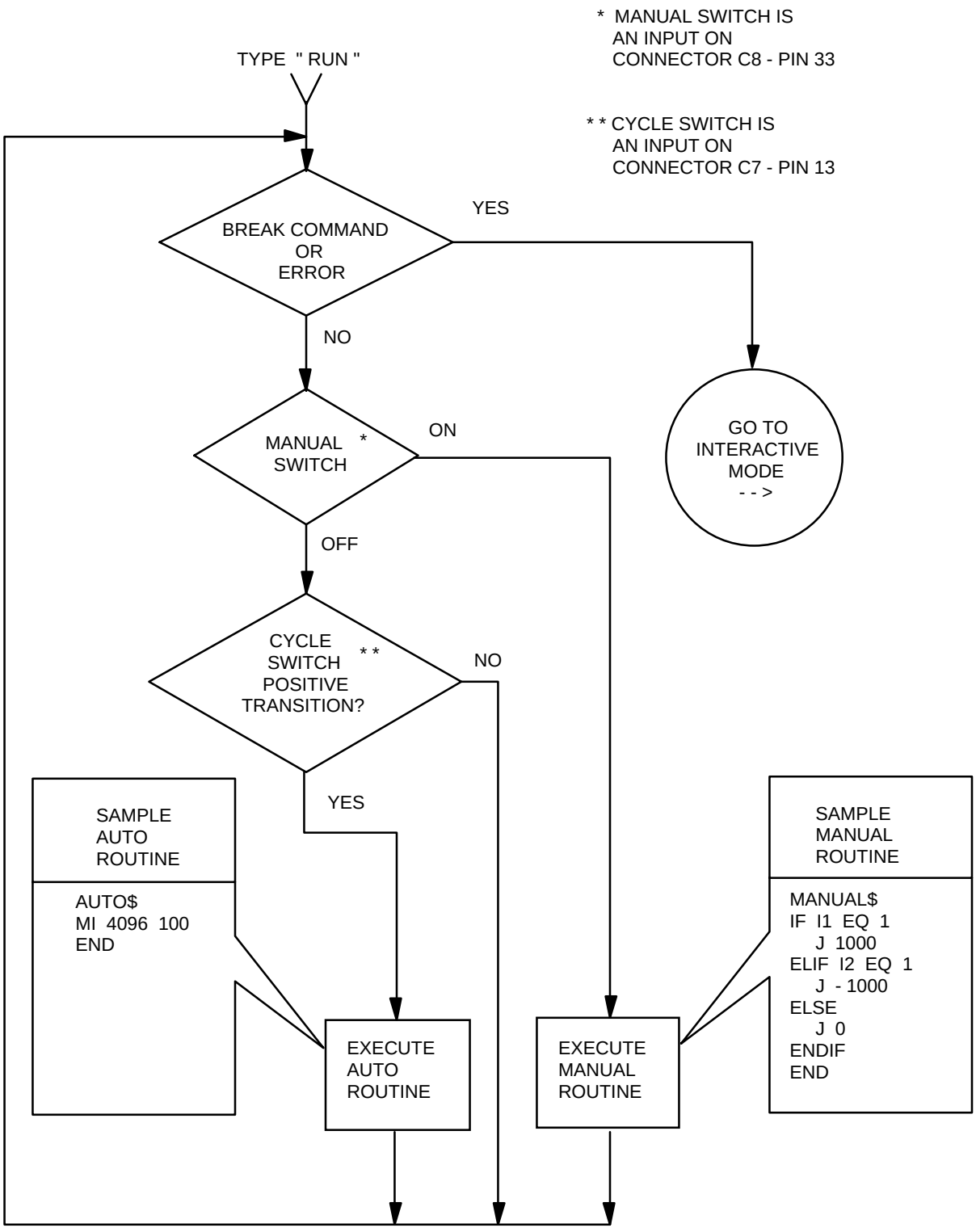


Figure 10.2 Auto/Manual Mode Flowchart

$$\text{ILIM [basic units]} = \text{ILIM [user units]} \times \frac{\text{INUM}}{\text{IDEN}}$$

INUM and IDEN have a range of 0 to 2^{31}. For standard current units (percent), INUM is 4095 and IDEN is 100. For example, when setting ILIM to 100 in Chapter 8, you typed:

```
ILIM=100          ;SET ILIM TO 100%
```

The VECTOR**STAR** converted the 100% to 4095 VECTOR**STAR**-basic units:

$$100 \times \frac{\text{INUM}}{\text{IDEN}} = 100 \times \frac{4095}{100} = 4095$$

This sets ILIM to 4095 or 100% of full current. When you typed:

```
P ILIM
```

the 4095 VECTOR**STAR**-basic units were converted to 100% by multiplying by IDEN and dividing by INUM.

10.9.1.2 Other User Units

VECTOR**STAR**-basic units for position, velocity, and acceleration vary with the system resolution. The resolution is determined by the R/D converter, which converts the position of the motor into a 12-, 14- or 16-bit number. The system resolution is indicated by the model number.

Table 10.7 System Resolutions

R/D Resolution	Counts in One Revolution
12-Bit	4096
14-Bit	16384
16-Bit	65536

When shipped from the factory, the standard VECTOR**STAR** user units are velocity in RPM, acceleration in RPM/second, current in percent of full-scale, and position in counts.

The velocity and acceleration units shown on "COMMON USER UNITS" above are for the standard 12-bit R/D converter. For 14-bit resolution, multiply VNUM and ANUM by 4. For 16-bit resolution, multiply by 16. Do not change VDEN or ADEN.

All variables that have units associated with them should be set after you have specified the user units. This is because the values actually stored in the variables are in VECTOR**STAR**-basic units, not user units. Changing the user units will not affect the basic value stored in the variables. For example, if you want VOSPD to be 100 inches/minute, and you type:

```
VOSPD = 100
```

When velocity units are in RPM, VOSPD would be 100 RPM. Then, if you change the velocity units to inches/minute, VOSPD would remain 100 RPM—it would just be converted to the equivalent of 100 RPM in inches/minute. If you change any user units, you should reset all programmable variables that depend on those units. Refer to Appendix F, which lists all variables and the units associated with them.

10.9.1.3 External Units

External units are for the external inputs, VEXT and PEXT. The user units are set by VXNUM and VXDEN for external velocity (VEXT) and by PXNUM and PXDEN for external position (PEXT). Figure 10.3 shows how external position and velocity come into a slave VECTOR**STAR** and are displayed as PEXT and VEXT.

If the external input is a system with the same resolution as your VECTOR**STAR**, set external units as follows:

VXNUM = VNUM
VXDEN = VDEN
PXNUM = PNUM
PXDEN = PDEN

If the command is something other than a motor of similar resolution, see "Machine Specific Units" in the next section.

10.9.2 Machine Specific Units

The VECTOR**STAR** allows you to specify user units for your machine. You must determine the conversion constants: PNUM & PDEN for position, VNUM & VDEN for velocity, and ANUM & ADEN for acceleration. Two tables have been provided to help you calculate those constants. Tables 10.8 and 10.9 are for position, velocity, and acceleration units based.

Table 10.8 English Conversion (12-bit R/D Only)

POSITION UNITS $\frac{PNUM}{PDEN} \mid 4096 \Delta \left\{ \frac{\text{Motor Movement / In Revolutions}()}{\text{Machine Movement / In Your Units}()} \right\}$
VELOCITY UNITS $\frac{VNUM}{VDEN} \mid 4473.92 \Delta \left\{ \frac{\text{Motor Velocity / In Rev / Min}()}{\text{Machine Velocity / In Your Units}()} \right\}$
ACCELERATION UNITS $\frac{ANUM}{ADEN} \mid 4.47392 \Delta \left\{ \frac{\text{Motor Acceleration / In RPM / Sec}()}{\text{Machine Acceleration / In Your Units}()} \right\}$

Table 10.9 Metric Conversion (12-bit R/D Only)

POSITION UNITS $\frac{PNUM}{PDEN} \mid 651.8971 \Delta \left\{ \frac{\text{Motor Movement / In Radians}()}{\text{Machine Movement / In Your Units}()} \right\}$
VELOCITY UNITS $\frac{VNUM}{VDEN} \mid 712.047 \Delta \left\{ \frac{\text{Motor Velocity / In Rad / Sec}()}{\text{Machine Velocity / In Your Units}()} \right\}$
ACCELERATION UNITS $\frac{ANUM}{ADEN} \mid 0.712047 \Delta \left\{ \frac{\text{Motor Acceleration / Rad / Sec / Sec}()}{\text{Machine Acceleration / Your Units}()} \right\}$

The procedure to determine PNUM and PDEN is as follows:

A. Select Table 10.8 (revolutions) or 10.9 (radians).

B. Select a convenient amount of motor movement in revolutions or radians.

C. Calculate the corresponding machine movement in your user units.

D. Perform the operation indicated in the table under POSITION UNITS and set PNUM/PDEN equal to this value.

E. If your R/D converter resolution is 14-bits, multiply PNUM by 4. Multiply PNUM by 16 for a 16-bit system.

The procedure to determine VNUM and VDEN is as follows:

A. Select Table 10.8 (RPM) or 10.9 (radians/second).

B. Select a convenient amount of motor velocity in RPM or radians/second.

C. Calculate the corresponding machine velocity in your user units.

D. Perform the operation indicated in the table under VELOCITY UNITS and set VNUM/VDEN equal to this value.

E. If your R/D converter resolution is 14-bits, multiply VNUM by 4. Multiply VNUM by 16 for a 16-bit system.

The procedure to determine ANUM and ADEN is as follows:

A. Select Table 10.8 (RPM/sec) or 10.9 (radians/(second*second).

B. Select a convenient amount of motor acceleration.

C. Calculate the corresponding machine acceleration.

D. Perform the operation indicated in the table under ACCELERATION UNITS and set ANUM/ADEN equal to this value.

E. If your R/D converter resolution is 14-bits, multiply ANUM by 4. Multiply ANUM by 16 for a 16-bit system.

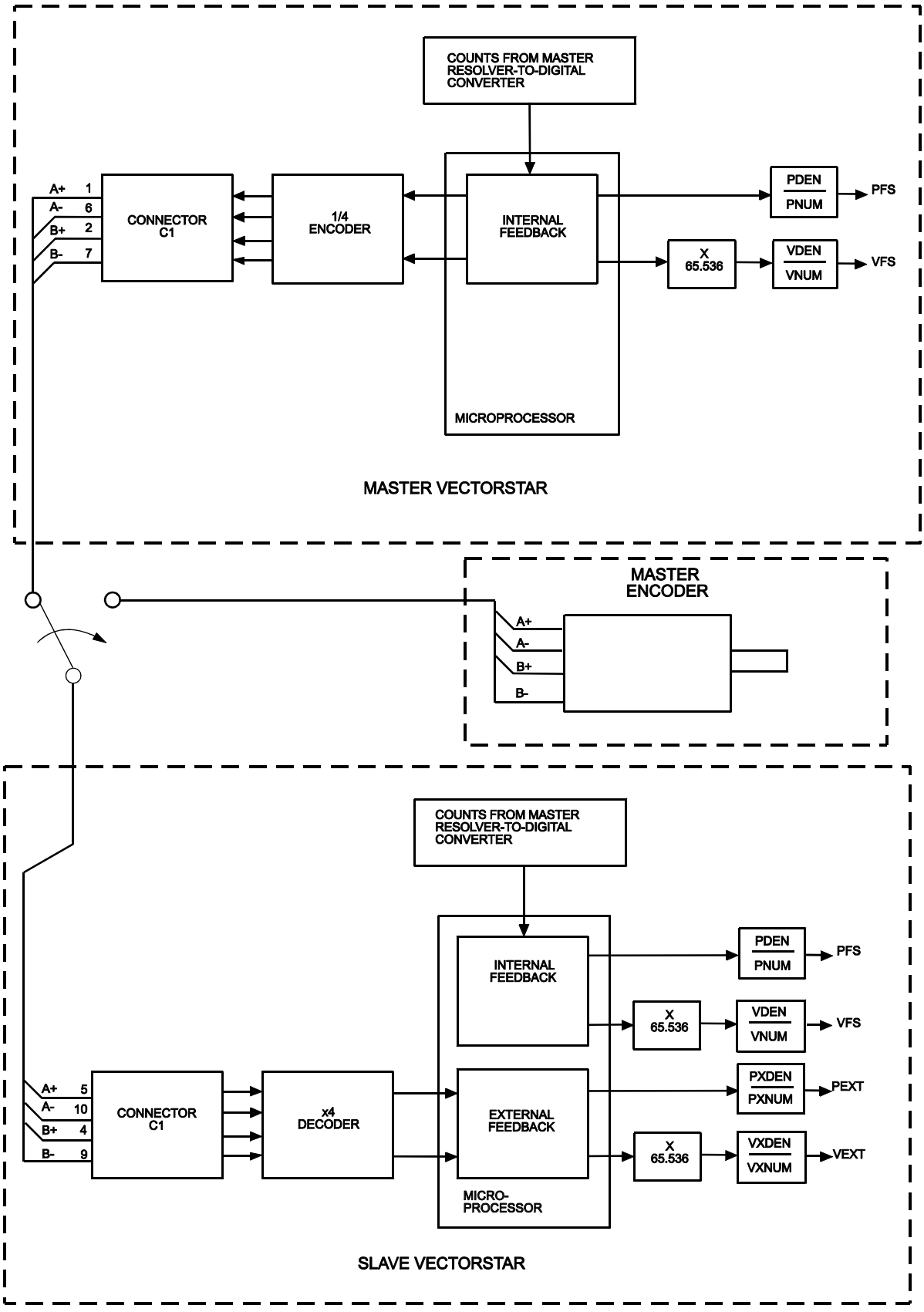


Figure 10.3 Master/Slave Block Diagram

For external inputs PEXT and VEXT, the procedure for calculating the conversion constants PXNUM, PXDEN, VXNUM, and VXDEN is similar. It differs in that the external inputs are not functions of the motor position or R/D resolution. Table 10.10 has been provided to assist in calculating the conversion constants.

Table 10.10 External Units Conversion

EXTERNAL POSITION UNITS $\frac{PXNUM}{PXDEN} \mid \left\{ \frac{\text{External Input (In Counts)}}{\text{Machine Movement /In Your Units}} \right\}$
EXTERNAL VELOCITY UNITS $\frac{VXNUM}{VXDEN} \mid 65.535 \Delta \left\{ \frac{\text{External Input (In Counts/Sec)}}{\text{Machine Velocity /In Your Units}} \right\}$

The procedure to determine PXNUM and PXDEN is as follows:

A. Select a convenient number of counts on the external input.

B. Calculate the corresponding machine movement in your user units.

C. Perform the operation indicated in Table 10.10 under EXTERNAL POSITION UNITS and set PXNUM/PXNUM equal to this value.

The procedure to determine VXNUM and VXDEN is as follows:

A. Select a convenient number of counts per second on the external input.

B. Calculate the corresponding machine velocity in your user units.

C. Perform the operation indicated in Table 10.10 under EXTERNAL VELOCITY UNITS and set VXNUM/VXNUM equal to this value.

Example:
A machine has a motor coupled to a 0.1 inch pitch lead screw which drives a table. A 0.1 inch pitch lead screw means the table moves 0.1 inch per motor revolution. The R/D resolution is 12 bits.

The user units for table motion you desire are:

Position Units	mils (1 mil = 0.001 inch)
Velocity Units	inches/minute (IPM)
Acceleration Units	inches/minute/second (IPM/second)

Objective:

Find PNUM and PDEN.
Find VNUM and VDEN.
Find ANUM and ADEN.

Solution:

Find PNUM and PDEN.

A. Select Table 10.8.

B. Select a motor movement of 1 revolution.

C. 1 revolution of the 0.1 pitch lead screw translates to 0.1 inch or 100 mils of table movement.

D. Refer to Table 10.10 under POSITION UNITS for the formula:

PNUM/PDEN = 4096 * (1 / 100) = 40.96

Select PNUM and PDEN:

PNUM = 4096 PDEN = 100

E. Since a 12-bit R/D converter is used, calculations in step E are not needed.

Find VNUM and VDEN.

A. Select Table 10.8.

B. Select 10 RPM motor velocity.

C. 10 RPM of the 0.1 pitch lead screw translates to 1 IPM of table velocity.

D. Refer to Table 10.11 under VELOCITY UNITS for the formula:

VNUM/VDEN = 4473.92 * (10 / 1) = 44739.2

Select VNUM and VDEN:

VNUM = 447392 VDEN = 10

Find ANUM and ADEN.

A. Refer to Table 10.8.

B. Select 10 RPM/second motor acceleration.

C. A 10 RPM/second acceleration of the 0.1 pitch lead screw translates to 1 IPM/second of table acceleration.

D. Refer to Table 10.8 under ACCELERATION UNITS for the formula:

ANUM/ADEN = 4.47392 * (10 / 1) = 44.7392

Select ANUM and ADEN:

ANUM = 447392 ADEN = 10000

The VECTOR**STAR** does not support floating point operations. You must use fractional units to make the resolution finer. For example, if the units for velocity need to be finer than IPM, 0.1 IPM could be chosen. In this case VDEN would be 100 instead of 10. Then to jog at 1 IPM the command J 10 would be required.

10.9.3 Position Rotary Mode: ROTARY and PROTARY

The VECTOR**STAR** stores position in a 32-bit number. This number is large enough to count many revolutions. For example, the 32-bit number will store the counts from a 12-bit R/D converter for about 10 million revolutions before the 32-bit limit is exceeded. Normally, this is sufficient. However, some applications require the motor to rotate in one direction indefinitely. Eventually, the 32-bit limit will be exceeded, resulting in an error. The *Rotary* mode allows the VECTOR**STAR** to support these unidirectional applications.

The Rotary mode forces all position-related variables to "roll-over" after position feedback (PFB) exceeds a specified limit. The variables that are rolled over are PFB, PCMD, and PFNL. The rotary distance (the specified limit before roll-over) is stored in PROTARY. PROTARY is in position units.

When ROTARY is on, the Rotary mode is enabled. If PFB is greater than PROTARY, then PFB, PCMD, and PFNL are decremented by PROTARY. If PFB is less than zero, then PFB, PCMD, and PFNL are incremented by PROTARY. Note that DIR = 0 does not work well with the Rotary mode as PCMD , PFB, and PFNL are always less than zero.

You cannot change PNUM, PDEN, or PROTARY when ROTARY is ON. In addition, you must normalize PFB so that 0 < PFB < PROTARY before turning ROTARY ON. Enable the Rotary mode by typing:

```
ROTARY ON
```

10.9.3.1 Choosing PROTARY, PNUM, and PDEN

If you have a rotary application such as a printing drum, set PROTARY in position user units to be the exact equivalent of one revolution of the drum. PROTARY must be exact or position error will accumulate over many revolutions. For example, suppose the motor of an application is connected through a 5:3 gearbox. For convenience, assume the user units are in degrees of the table. PROTARY would be one revolution of the table or 360 degrees. How do you select PNUM, PDEN, and PROTARY?

The key is selecting PNUM and PDEN so that PROTARY can be represented exactly as an integer. This does not mean that PROTARY must be an integer number of counts. In fact, it normally will not be. Returning to the example, a motor movement of 5 revolutions would cause 3 revolutions of machine (table) rotation, or 1080 user units (degrees). Returning to Table 10.9,

PNUM = 4096 * 5 PDEN = 360 * 3

thus, PROTARY would be 360. Notice that PROTARY is not exact in counts; it is 5/3 of a revolution or 6826 and 2/3 counts. However, it is exact in user units. Therefore, error will not accumulate as the table rotates.

The incorrect way to choose PNUM, PDEN, and PROTARY would be to select PNUM and PDEN so that PROTARY could not be represented as an integer. For example, we could have stated that 5/3 revolution of the motor would cause one revolution of the machine. Then:

PNUM = 4096 * 5/3 or about 6827

PDEN = 360

In this case, PROTARY would not be exactly 360 degrees (actually, it would be 359.98 degrees), so that error would accumulate as the table turned. Remember, PROTARY must be an integer in user units, though it can have fractional counts.

10.9.3.2 Rotary Mode and Absolute Moves

When the VECTOR**STAR** is in the Rotary mode, you must limit the final position of all absolute moves to between 0 and PROTARY. If you want to move more than PROTARY, you can use incremental moves. For example,

```
MI 50*PROTARY
```

is a legal command.

10.10 SERIAL COMMUNICATIONS

This section discusses details of VECTOR**STAR** serial communications. This includes autobauding, multidrop connections, and transferring your program to and from the VECTOR**STAR**. If you are using Motion Link, the Kollmorgen software package for the VECTOR**STAR**, you do not need to read the sections on transmitting and receiving your program, or on system dump. Motion Link provides facilities for these functions.

10.10.1 Autobauding

It is not necessary to set the baud rate on the VECTOR**STAR** directly. Once the VECTOR**STAR** is properly connected, it can determine the terminal's baud rate, then set its own baud rate accordingly. This is called autobauding. After the VECTOR**STAR** determines the correct baud rate, it will store this rate away in BAUD. The VECTOR**STAR** will flash the CPU light to indicate that it is autobauding. In order for the VECTOR**STAR** to determine the baud rate setting on your terminal, you must press the enter key several times. Press only the enter key; otherwise the VECTOR**STAR** will not autobaud correctly. The system will only autobaud during power-up.

10.10.1.1 Setting the VECTORSTAR to Autobaud

There are three ways to set the VECTOR**STAR** to autobaud at power-up:

1. Powering-up with the MOTION input off.

2. Turning the switch ABAUD on before the next power-up.

3. Setting the value of the variable BAUD to an invalid value (1000, for example).

10.10.1.2 Autobauding and MOTION

If the MOTION input is off during power-up, the VECTOR**STAR** will autobaud. (Note that this also sets ADDR to zero.) This allows you to command autobaud without being able to communicate with the VECTOR**STAR**. The other ways to start autobauding require that communications be set up first. See the section on ADDR and multidrop communication later in this chapter for more information.

10.10.1.3 Enabling Autobaud with ABAUD

The autobaud software switch (ABAUD) is the usual way to tell the VECTOR**STAR** to autobaud on power-up. If ABAUD is on, then the system will autobaud when it is powered-up or reset, provided that the multidrop address, ADDR, is 0. After a successful autobaud, the baud rate will be stored in BAUD.

If you do not want your VECTOR**STAR** to autobaud when the unit is powered-up, then turn ABAUD off. This is important if you want the VECTOR**STAR** to run the Power-Up Label (POWER-UP$), because if ABAUD is on, the VECTOR**STAR** will not execute the program until communications have been established.

10.10.1.4 Baud Rate, BAUD

If the MOTION input is on, ADDR is zero, and ABAUD is off, then the system will check the variable BAUD for the desired baud rate. If it is not a valid baud rate, the VECTOR**STAR** will autobaud. After a successful autobaud, an error is generated indicating that the baud rate was out-of-range on power-up.

10.10.2 Prompts

The VECTOR**STAR** issues a prompt when it is ready to receive a new command. Prompts are discussed in Chapter 8. The VECTOR**STAR** allows you to suppress the prompt characters by typing:

```
PROMPT OFF
```

PROMPT is turned on at power-up. Prompts are particularly important when communicating with computers, since the computer that is transmitting to the VECTOR**STAR** must wait for a prompt before beginning a new line. After the prompt is received, the computer can transmit at the full baud rate, without inserting delays.

10.10.3 Serial Watchdog

The VECTOR**STAR** provides a serial watchdog timer for applications where a command should be received from a computer on a regular basis. If a complete command is not received from the serial port in the specified time, an error will be generated that will disable the VECTOR**STAR** and break the user program.

The serial watchdog is a safety feature that disables the VECTOR**STAR** if the communications line breaks. The serial watchdog waits for a carriage return to signify a completed command. It does not test the validity of the command. For example, if your computer fails and begins sending random carriage returns, the serial watchdog will not generate an error.


WARNING

The VECTORSTAR serial watchdog is intended to detect a broken serial communications line. It does not test the validity of data received from your computer.

Set WTIME in milliseconds to the time that you want the serial watchdog to timeout. To enable the serial watchdog, type:

WATCH ON

10.10.4 Transmit/Receive Programs

The VECTOR**STAR** provides commands that allow programs to be transmitted and received without using the Editor. These commands are intended for applications which require that a computer directly transmit and receive programs. This does not include Motion Link, the software communications package that is run from an IBM-PC or compatible. Refer to Chapter 8 for communications format.

10.10.4.1 <BDS Command Receiving from the VECTORSTAR

The <BDS command is used to send the VECTOR**STAR** user program through the serial port to the terminal or computer. The transmission can be stopped by sending an escape character. You should not rely on the VECTOR**STAR** to store all your programs. Keep back-up copies elsewhere. The <BDS command will cause the VECTOR**STAR** to transmit the entire user program to your computer. It cannot be issued in the Program mode. For example, if you type the following command from the terminal, the VECTOR**STAR** will respond by printing out the entire user program.

<BDS

10.10.4.2 The >BDS Command Transmitting to the VECTORSTAR

The >BDS command is used to send a new user program through the serial port to the VECTOR**STAR**. The transmission is ended by sending an escape character. Note that this command writes over the contents of the user program stored in the VECTOR**STAR**. This command allows the program to be directly entered, presumably by a computer, to the VECTOR**STAR**. It cannot be issued in the Program mode.


NOTE

The >BDS command writes over the entire user program.

The VECTOR**STAR** issues the "l->" prompt to indicate that it is ready to load a new program line. If you are loading from a computer, you must wait for this prompt before beginning to transmit a new line.

The >BDS command is password protected. If a password was set in the VECTOR**STAR** Editor, then it must be given in the >BDS command.


NOTE

Typing in these examples will erase the user program in the VECTORSTAR. Do not type them in unless your program is backed up.

For example, if a password was not set in the Editor,

>BDS

will begin transmitting the new program. If you press the escape key before typing anything else, the process will be aborted without changing the program in the VECTOR**STAR**.

If a password was set in the Editor, then the password must follow the command. For example, if the password was set as SECRET, type:

```
>BDS  SECRET
```

and the VECTOR**STAR** will accept programs directly from the terminal.

The user program is stored in battery backed-up memory. If the program changes because of a hardware problem, the VECTOR**STAR** issues a "USER PROGRAM CORRUPT" error. The >BDS command resets the user program memory, which eliminates this condition.

10.10.5 System Dump

The VECTOR**STAR** can transmit all variables in addition to the user program. This is called a system dump, and you request it with the DUMP command. For example, type:

```
DUMP
```

and the VECTOR**STAR** will provide pages of information including the program, all VECTOR**STAR** variables, user variables, and user switches. This also includes all protected variables.

The system dump is provided so that the information from the dump can be directly re-transmitted to any VECTOR**STAR**. This changes all *NON-PROTECTED* variables. The DUMP command precedes protected variables with a semicolon (;). This makes the line a comment so that when the line is re-transmitted, it has no effect. If the semicolon were not there, re-transmitting the dump information would generate an error when a protected variable was changed. Every line of the user program is preceded with a semicolon for the same reason.

10.10.5.1 Version Dump

Your VECTOR**STAR** will print out its firmware version at any time with the DUMP VERSION command:

```
DUMP  VERSION
```

10.10.6 Multidrop Communications

This function is not available for the RS-232 option.

Multidrop communication allows you to have many (up to 32) axes on one serial line. This is only supported with RS-485. When the VECTOR**STAR** is in Multidrop mode, each axis must have a unique address. This address is a prefix on all communications to and from the VECTOR**STAR**. The address is stored in variable ADDR. ADDR is set to 0 for standard (single-drop) communications. Valid addresses are 48 (ASCII '0') through 57 (ASCII '9') and 65 (ASCII 'A') through 90 (ASCII 'Z') (see Appendix B). Note that the address must be set before multiple units are connected to the same serial line.

When the VECTOR**STAR** powers-up in Multidrop mode it is "asleep." When asleep, the VECTOR**STAR** continues to execute programs and control the motor properly, but it does not communicate over the serial line. The VECTOR**STAR** executes commands which normally print to the serial port (P, PS, R, RS, INPUT, and errors) except that the output is not sent to the serial transmitter. The delays incurred by printing are still present. If you have print statements that delay the program when the axis is awake, you will have the same delays when it is asleep, even though no characters are being transmitted.

When you transmit its address, the VECTOR**STAR** wakes up and communicates. The address is a backslash (\) followed by the ASCII character represented by ADDR. For example, if your VECTOR**STAR** has the RS-485 option, type:

```
ADDR=65              ;SET ADDRESS TO
                     ;65=ASCII A
\A                   ;WAKE UP "A"
P  "THIS IS AXIS"  ADDR
                     ;PRINT ADDR
ADDR=0
                     ;RESET DRIVE TO
                     ;SINGLE-DROP
```

This example sets the address to upper case A.

Setting ADDR to 65 makes this axis address "A" and automatically puts the VECTOR**STAR** in Multidrop mode. This axis then waits for the \A. After this, VECTOR**STAR** is awakened and it remains awake until it receives a backslash (\). A backslash puts ALL drives on the serial line to sleep. If you select an axis in multidrop, only that axis transmits and receives.

During multidrop, the prompts are changed. If you typed in the example from above, you would have noticed the prompt going from --> to A-> after you typed in the second line. All prompts in a multidrop system have the axis address as the first character of the prompt. This tells you which axis you are communicating with at all times. For example, the edit prompt goes from e-> to Ae>. In this way, each prompt from each axis is unique.

Table 10.11 VECTORSTAR Prompts

Non-multidrop (ADDR=0)	Multidrop (ADDR = 65)
-->	A->
==>	A=>
s->	As>
t..	At.
e->	Ae>
i->	Ai>
f->	Af>
c->	Ac>

10.10.6.1 Broadcast

You may want to send a command to all VECTOR**STAR**s on the serial line simultaneously. This is called a broadcast. You can broadcast by sending *. In this case, all VECTOR**STAR**s execute the command. During a broadcast, none of the VECTOR**STAR**s can transmit but all will receive and execute the command.

10.11 PROGRAM EXAMPLES

This section lists a typical application program as well as a sample velocity drive program. Use these programs as models for your own. This format uses extensive comments. The assumption is that you are using Motion Link so that these comments will not be transmitted to the VECTOR**STAR**, as they would normally take an unacceptable amount of space. You are encouraged to use comments because they make the program easier to understand and correct.

For the velocity drive program, you must first select whether the input will be analog or digital (encoder equivalent). Be sure to set GEARI and GEARO for your application.

```
;
;NAME OF APPLICATION:  PRETZEL
MACHINE
;
;DATE                    A.E. NEUMAN
;
;REVISION HISTORY:
;       8-9-90   ADDED JOG BUTTONS
;       7-17-90 CORRECTED TEACH BUG
;
;
;_____________________________________
_______-
;ALARM DESCRIPTION
;
;       A$                      WATCH
THERMOSTAT
;       B$,C$                   NOT USED
;       VARIABLE$               FILL X1 WITH
SPEED
;       BACKGROUND$             BACKGROUND
PRINTING
;
;_____________________________________
_______-
;
;I/O DESCRIPTIONS
;
;GENERAL PURPOSE INPUTS
;       I1      JOG+ PUSH BUTTON
;       I2      JOG- PUSH BUTTON
;       I3      TEACH POSITION PUSH
BUTTON
;       I4      CONTACTOR INTERLOCK
SWITCH
;       I5      PLC INTERFACE
;       I6      HOME REQUEST PUSH
BUTTON
;       I7      THERMOSTAT
;
;GENERAL PURPOSE OUTPUT
;       O1      COOLING FLUID PUMP
;       O2      SPINDLE MOTOR CONTACTOR
;       O3      PLC INTERFACE
;
;DEDICATED I/O
;       CYCLE CONNECTED TO PLC
```

```
;       GATE            NOT USED
;       HOME            CONNECTED TO HOME LIMIT SWITCH
;       LIMIT           CONNECTED TO OVERTRAVEL LIMIT SWITCH
;       MANUAL          NOT USED
;       MOTION          CONNECTED TO STOP PUSH BUTTON
;       READY           CONNECT TO PLC
;       STATUS          NOT USED
;
;————————————————————————————————————————————-
;
;USER VARIABLES
;       X1      STORE NUMBER OF CYCLES RUN
;       X2      STORE LAST POSITION RUN TO
;       X3      INTERMEDIATE CALCULATION
;       X4      LOOP COUNTER
;       X5      LOOP COUNTER
;       X6-X250 NOT USED
;       ;
;USER SWITCHES
;       XS1-XS50        NOT USED
;
;————————————————————————————————————————————-
;
;APPLICATION PROGRAM
;
POWER-UP$                       ;POWER-UP LABEL
PLIM  OFF                       ;SOFTWARE LIMITS NOT USED HERE
;CONTINUE YOUR POWER-UP PROGRAM HERE
END
;
;
A$  I7  OFF
P  “THERMOSTAT (INPUT I7) OPENED”
P  “PROCESS BEING CLOSED DOWN”
DIS                             ;DISABLE THE VECTORSTAR
B                               ;BREAK PROGRAM EXECUTION
;
;
VARIABLE$
INPUT  “ENTER NEW SPEED”  X1
END
;
;
AUTO$                           ;AUTO LABEL
;WRITE YOUR AUTO PROGRAM HERE
END
;
;
MANUAL$                         ;MANUAL LABEL
;WRITE YOUR MANUAL PROGRAM HERE
END
;

. . .
;WRITE MORE OF YOUR PROGRAMS HERE
```

```
END
;
BACKGROUND$
;WRITE YOUR BACKGROUND PRINTING ROUTINE HERE
END
;
;
ERROR$                  ;ERROR HANDLER
;WRITE YOUR ERROR HANDLER HERE
B                       ;END OF SAMPLE PROGRAM
```

```
;VELOCITY DRIVE SAMPLE PROGRAM
;DATE                   NAME
;————————————————————————————————————————————————————————————-

POWER-UP$               ;EXECUTE ON POWER UP
PL OFF                  ;DISABLE THE POSITION LOOP

VNUM=447392             ;SETS VELOCITY UNITS TO RPM.
VDEN=100

ANUM=447392             ;SETS ACC UNITS TO RPM/SEC
ADEN=100000
;
AMAX=100000             ;SET THE MAX ACCEL RATE (RPM/SEC)
ACC=1000                ;SET THE NORMAL ACCEL LIMIT
DEC=1000                ;SET THE NORMAL DECEL LIMIT
;ACC AND DEC ARE RAMP LIMITS FOR GEAR MODE,
;ASSUMING THAT PL IS OFF.
;
;
GEARI=10                ;THIS SETS THE GEAR MODE FOR 25%,
GEARO=40                ;APPROX. 10 V = 3000 RPM FOR AN
                        ;ANALOG INPUT.  THE PROPER LEVEL OF
                        ;GEARI AND GEARO DEPENDS ON THE
                        ;SYSTEM AND THE INPUT FORMAT.  THE
                        ;ADJUSTMENT OF GEARI AND GEARO IS
                        ;EQUIVALENT TO A DC GAIN ADJUSTMENT OR
                        ;SCALE FACTOR POT FOUND ON MANY
                        ;ANALOG DRIVES.

;NOTE THAT ACC/DEC RATES ARE LIMITED BY ACC AND
;DEC ONLY WHEN PL IS OFF.
;
EN                      ;ENABLE DRIVE
GEAR  ON                ;ENABLE ELECTRONIC GEARBOX
```

```
VOFF=0                  ;THIS SETS THE OFFSET VELOCITY.
                        ;VOFF IS SET TO ZERO WHEN GEAR IS
                        ;TURNED ON.
;IF THERE IS NEED TO ADJUST FOR VELOCITY
;DRIFT IN THE INPUT, THEN ADJUST VOFF
;TO THE PROPER LEVEL SO THAT DRIFT STOPS.
;
B                       ;DRIVE IS NOW IN ELECTRONIC GEARBOX
                        ;END OF SAMPLE PROGRAM
```

CHAPTER 11
DEBUGGING

11.1 INTRODUCTION

The information in this chapter will enable you to rectify problems you may have while programming the VECTOR**STAR**. When you write programs, you may inadvertently include a few errors or *bugs*. The best step you can take to correct errors is to prevent them by following the programming practices provided in this manual. Every effort has been made to make the VECTOR**STAR** language as simple as possible with BASIC-like commands, algebraic math, and a variety of conditional commands. Still, some bugs are almost certain to surface in a new program. The VECTOR**STAR** provides two execution modes to help you debug your program: Trace and Single-Step.

11.2 DEBUGGING MODES

11.2.1 Single-Step (SS)

If the error occurs in a section of your program that is not time-critical, you can use single-stepping to help track down the error. When you execute your program in the Single-Step mode, each command is printed out. The VECTOR**STAR** waits for you to press the ENTER key before executing the command. Use the nested-IF example given in Chapter 10. Enter the program, set X1 and X2 equal to 1, and turn SS on by typing *SS ON*. Then begin execution at label 55 by typing *RUN 55*. The following line should be displayed:

```
55$
S—>
```

Press the ENTER key and the response should be:

```
IF X1 GT 0
S—>
```

You can probe the VECTOR**STAR** variables from the Single-Step mode without stopping your program. For example, type:

```
P X1
```

and the VECTOR**STAR** should respond with:

```
        1
S—>
```

In this case, the VECTOR**STAR** executed the print command and displayed the single-step prompt, indicating it is ready for another command. Now press the ENTER key repeatedly to step through the program.

This example shows several characteristics of the Single-Step mode:

- All commands are preceded by the trace prompt:

```
S->
```

- Print statements are active in the Single-Step mode. Notice that the results of the P command are printed normally, as they are in the Trace mode.

- Only the executed commands in the IF, ELIF, ELSE, and ENDIF sets are shown. Notice that none of the commands following the first print command are shown.

- You can execute commands from the Single-Step mode.

You can also enter the Single-Step mode from your program. To do this, you should include *SS ON* in your program. To exit the Trace mode, you can include *SS OFF* in your program or type it from the single-step prompt. You can also press the escape key two times.

11.2.2 Trace (TRC)

If the error occurs in a section of your program that is not very time-critical, you can use trace to help track down the error. When you execute your program in the Trace mode, each command is printed out just before it is executed. Use the nested-IF example given earlier in this chapter. Enter the program, set X1 and X2 equal to 1, and turn TRC on (*TRC ON)*. Then begin execution at label 55 (*RUN 55)*, and the following lines should be displayed:

```
T...55$
T...IF X1 GT 0
T... IF X2 GT 0
T...  P "BOTH X1 AND X2 > 0"
BOTH X1 AND X2 > 0
T...  ELSE
T...  ENDIF
T...ELSE
T...ENDIF
T...B
—>
```

This example shows several characteristics of the Trace mode:

- All commands are preceded by the trace prefix:

```
T...
```

- Print statements are active in the Trace mode. Notice that the results of the P command are printed just below where the print command is displayed.

- Only the executed commands in IF, ELIF, ELSE, and ENDIF sets are shown. Notice that none of the commands following the first print command are shown. This helps you debug your program by only showing the commands that are executing.

- You cannot type in commands from your terminal while the VECTOR**STAR** is executing in the Trace mode.

You can also enter the Trace mode from your program. To do this, you should include *TRC ON* in your program. To exit the Trace mode, you can include *TRC OFF* in your program, or you can press the escape key two times.

11.2.2.1 Motion Link and Trace

Motion Link is the software communications package provided for the IBM-PC and compatibles. IBM-PC and compatibles can communicate at 9600 baud only in that they can receive and transmit a character at that frequency. However, they cannot receive an indefinite number of characters at that rate because the computers are not fast enough to process the characters. This leads to a problem in the Trace mode because the VECTOR**STAR** can transmit characters much faster than most PC's can process them. This can lead to delays of minutes between when the VECTOR**STAR** transmits a character and when the computer displays it. The best way to cure this problem is to reduce the baud rate from Motion Link (use the ^U command), and power the VECTOR**STAR** down and then up to cause a second autobaud (make sure ABAUD is on before powering down). Start with 1200 baud and see if the problem is cured.

11.3 DEBUGGING AND MULTI-TASKING

If your program uses multi-tasking, the Trace and Single-Step modes show you which level is currently being executed. For example, enter the program given in Section 10.8.5.2. Turn on the Trace mode and type:

```
RUN 1
```

The result should be something like this:

```
T...1$                  ;MAIN PROGRAM
T...EN
T...MI  10000  10       ;START MOVE
T...P  "MOVE PROCESSED"
MOVE PROCESSED
T...W  0                ;WAIT FOR MOVE
T.*.BACKGROUND$
T.*.P  "UPPER TASK IDLED"
UPPER TASK IDLED
T.*.D  250              ;DWELL 0.25 SEC.
T.*.END

...

T.*.BACKGROUND$
T.*.P  "UPPER TASK IDLED"
UPPER TASK IDLED
T.*.D  250              ;DWELL 0.25 SEC.
T.*.END

(AT THIS POINT, ASSUME MOTION STOPS
AND TASK 5 IS NOT IDLED)

T...P  "ALL MOTION STOPPED"
ALL MOTION STOPPED
T...B
```

Notice that when the example is executing the background level task, an asterisk (*) is printed. Each task level prints out a slightly different prompt in the Trace and Single-Step modes, as the following table shows:

Table 11.1 Multi-Tasking Debug Prompts

TASK LEVEL PROMPT	SINGLE-STEP PROMPT	TRACE PROMPT
Alarm A	s-A>	t.A.
Alarm B	s-B>	t.B.
Alarm C	s-C>	t.C.
Variable Input	s-V>	t.V.
Main Program	s-->	t...
Background	s-*>	t.*.

11.4 REMOVING CODE

If you cannot find the bug in your program with single-step or trace, then you must begin removing sections of your code that you do not think are causing the problem. The procedure is to remove sections of your program a few lines at a time. (Of course, save the original program on your computer for later use.) Remove lines that you do not think are involved in the problem. Removing lines that you suspect are causing the problem can provide false leads; for example, the problem may be interaction between a section you removed (which was operating properly) and another, unsuspected section of your program (that was the actual source of the problem). Your false suspicions can be incorrectly confirmed.

The best situation is when you can make a short (< 20 line) program demonstrate the problem. After this, it is usually easy to determine the problem. If you get to the point where you cannot figure out your problem, call Kollmorgen; we will be happy to help you. However, in order to make efficient use of your time and ours, you must trim down your program to a few lines that are not working. It is very difficult for even a skilled person to help debug a large program over the telephone.

11.5 SYNCHRONIZING YOUR PROGRAM

This section describes the functions and variables that allow you to synchronize the program to events, both external and internal.

11.5.1 Using the Timers, TMR1-4

The general purpose timers TMR1, TMR2, TMR3, and TMR4, are provided for situations where the required timing is too complex for the DWELL command. The timers are set in milliseconds and are limited to 2,147,483,647 milliseconds or about 25 days. The VECTOR**STAR** then counts down the timer until it reaches zero.

Type in this example, which continuously reprints a message for 1 second:

```
8$
TMR1=1000
TIL  TMR1  LE  0  P  "WAITING FOR 1
SECOND DELAY"
B
```

and type:

```
RUN 8
```

Type in this example showing how multiple waits can be based on one timer setting:

```
9$
TMR1  3000   ;SET TMR1 TO 3 SECONDS
P  "3 SECONDS"
TIL  TMR1  LE  2000
P  "2 SECONDS"
TIL  TMR1  LE  1000
P  "1 SECOND"
TIL  TMR1  EQ  0
B
```

and type:

```
RUN 9
```

11.5.2 Regulation Timer, RD

Fixed length delays can be added into a program with the DWELL (D) command. In some applications, especially those that use profile regulation, it is necessary to add a delay with a length that varies with the regulating frequency. The DWELL (RD) command is provided for these occasions. When the external input frequency is equal to REGKHZ the delay of the RD command is in milliseconds, just like D command. However, when the external input frequency decreases, the regulated dwell time lengthens so that the DWELL is proportional to the inverse of the external frequency. For example:

```
45$
REGKHZ  100          ;SET REGKHZ TO
       ;100 KHZ
RD  2000             ;REG DOES NOT
                     ;NEED TO BE ON
                     ;FOR RD TO OPERATE
P  "DELAY COMPLETE"
B
```

In this case, the RD command causes a 2-second dwell when the external input frequency is 100 KHz and a 4-second dwell when the frequency is 50 KHz. Note that MACRO DWELLS (MCD) are regulated by the external input when REG is on. RD delays are always regulated by the external frequency, even when REG is off.

11.5.3 Motion Segments

All moves and jogs occur in segments. Normal jogs have two segments: accel/decel, and traverse. Simple moves (MRD, MI, and MA) have three segments: accel, traverse, and decel. Position dependent jogs have three segments: traverse to position, accel/decel, and traverse. The following table shows the different segments for VECTORSTAR moves:

Table 11.2 Segments for Different Moves

Segment	MI,MA,MRD	J	JT/JF
1	Accel	Accel/Decel	Traverse
2	Traverse	Traverse	Accel/Decel
3	Decel	N.A.	Traverse

Macro moves have up to 30 segments, where each accel, decel, traverse, and dwell counts as a segment. In each case, every move begins with the variable SEG equal to 1. As the move progresses, SEG is incremented. When all moves are complete, SEG is set to zero.

You can use the SEG to determine when motion is complete, since SEG is zero when the VECTOR-**STAR** is not commanding a profile. The following example continually prints a message until motion stops.

```
46$
MA  10000  1000
TIL  SEG  EQ  0   P  "MOTION IN
PROGRESS"
B
```

Note that when SEG is zero, the VECTORSTAR is not commanding motion. However, because there is a lag between the command and the response of the motor, you may want to insert a short delay after SEG is zero:

```
46$
MA  10000  1000
TIL  SEG  EQ  0   P  "MOTION IN
PROGRESS"
D  100          ;DWELL 100 MSEC—WAIT
                ;FOR MOTION TO SETTLE
                ;OUT.  AT THIS POINT
                ;MOTION SHOULD BE
;ZERO
B
```

The commands *TIL SEG EQ 0* and *W 0* are similar, since both delay execution until motion profiles are complete. However, the *W 0* command is an idling command and thus allows lower level tasks to execute. Also, the TIL command can be followed with a statement (such as the P command above), which is executed continuously until motion stops. If you want to synchronize to a segment, the SEG variable can be used with the TIL command. For example, suppose you want to turn on an output after the decel of an MI move begins. The following sequence can be used:

```
47$
O1  OFF              ;TURN OFF OUTPUT 1
MI  -50000  1000     ;BEGIN THE MOVE
TIL  SEG  EQ  3      ;WAIT HERE UNTIL
                     ;SEGMENT 3 IS
                     ;STARTED
O1  ON               ;TURN ON OUTPUT 1
B
```

11.5.4 WAIT (W)

The WAIT (W) command can also be used for synchronization. The WAIT command is *W* followed by the segment for which you want the program to wait, or a 0 if you want the program to wait for motion to stop. WAIT is provided in addition to the TIL command because it takes less space in your program. For example, *W 3* performs a similar function to *TIL SEG EQ 3*.

The WAIT command provides a few special features needed for motion synchronization. For example, in the following program, the Wait delays execution until segment 2 of the second move.

```
MI  -50000  1000     ;BEGIN THE FIRST
                     ;MOVE
MI  -50000  1000     ;CALCULATE THE
                     ;SECOND MOVE
W  2                 ;WAIT FOR SEG 2
                     ;OF THE SECOND
                     ;MOVE
```

If *TIL SEG EQ 2* were used in place of *W 2*, then execution would delay until segment 2 of the first move. Since you normally want to wait for the specified segment of the last move calculated, the WAIT command always applies to the last move.

The WAIT command never waits when motion has stopped. For example, if you entered the following program the TIL command would delay execution indefinitely because SEG would never equal 4.

```
MI  -50000  1000
TIL  SEG  EQ  4      ;BUG—DELAYS
                     ;INDEFINITELY
```

However, the following program only delays until motion stops because the WAIT command does not delay program execution when motion has stopped.

```
MI  -50000  1000
W  4                 ;BUG—DELAYS
                     ;UNTIL MOTION
                     ;STOPS
```

Normally, you should use the WAIT command when you are synchronizing motion to program execution. It is an idling command and thus allows lower level tasks to execute; also, it takes less space, waits for the last motion program, and it does not delay execution when motion has stopped. Use the TIL command when you need a special function, such as printing during the wait, or if you specifically want to stop lower level tasks from executing.

Another example of the WAIT (W) command is seen when using multiple JOG TO/JOG FROM commands. Normally, you should place a WAIT (W) command between these commands, because the initial traverse of a JOG FROM/JOG TO command begins as soon as the command is entered. Usually, you will want the traverse to begin at the end of the last specified acceleration segment. For example, consider Macro Move Example #1 in Chapter 8. It could have been done with one JOG and two JOG TO commands:

```
J  1000              ;START MOTION
W  2                 ;WAIT TIL JOG
                     ;ACCEL IS DONE
JT  10000  200       ;ENTER JT FOR
                     ;FIRST DECEL
W  3                 ;WAIT TIL JT DECEL
                     ;IS DONE
JT  11000  0         ;ENTER FINAL
                     ;SEGMENT OF MOVE
```

11.5.5 Gating Motion with GATE

The GATEMODE variable allows you to pre-calculate a profile and begin motion within 1.5 milliseconds of a switch closure. To enable GATE, turn on GATEMODE and follow it with either:

1. One or two MA or MI commands,
2. One or two Macro Go (MCGO) commands, or
3. One Jog or MRD command.

When the hardware input GATE transitions from low to high, motion begins. GATE is on Connector C7, Pin 17. After motion is begun, GATEMODE is turned off. You must re-enable GATEMODE for each move that you want gated. Also, you cannot turn GATEMODE on when motion is commanded from Jogs, MA, MI, or

MCGO commands. If you turn GATEMODE on and command motion, but turn GATEMODE off before the GATE input turns on (thus, allowing motion to begin), the commanded motion will be "forgotten" by the VECTOR**STAR**.

In the following example, two MI commands are entered and precalculated with GATEMODE on.

```
GATEMODE  ON      ;ENABLE GATING
MI  1000  100     ;PRECALC MOVES.
                  ;MOTION DELAYED
MI  -1000         ;MOTION DELAYED
                  ;TIL GATE IS HIGH
W  0              ;WAIT FOR MOTION
                  ;TO START
```

This means no motion will take place until the hardware input GATE is high. If the above lines were part of a program, the W command would delay program execution until the GATE switch was on.

11.6 HINTS

The following section lists some hints addressing the most common problems. Most result from a minor misuse or misunderstanding of a VECTOR**STAR** function.

If you change your program in the Motion Link Editor and the program function does not change, you may have forgotten to transmit your updated program to the VECTOR**STAR**.

If you command motion with MI, MA, MCGO, J, JT, or JF, and the motor does not move...

...make sure GATEMODE is not preventing motion (turn GATEMODE off if you are not certain).

...make sure CLAMP is not preventing motion (turn CLAMP off if you are not certain). If it is CLAMP, try raising the clamp limit, PECLAMP, somewhat. If that does not help, turn CLAMP off. If you now get PE OVERFLOW errors, it may be because the motor is undersized. See the hints for PE OVERFLOW errors below.

...make sure REG is not preventing motion (turn REG off if you are not certain). If REG is on, you may not be feeding in the master encoder signal properly. Remember, it must always count up. Check VEXT. It should be greater than zero for profile regulation to work.

...make sure ZERO is off.

...make sure all tuning constants are well above zero. Check KP, KV, KVI, and KPROP. Each should be at least one hundred; generally, they are above one thousand.

...make sure ILIM is not too small. If ILIM is below 10%, the motor may not be able to overcome frictional load.

...make sure you are commanding a speed that you can see. The VECTOR**STAR** can command speeds as low as .0004 RPM or about one revolution every three days, depending on how you program velocity units. If you have changed VNUM or VDEN from the factory setting, temporarily restore them to see if the problem goes away.

If the motor moves and you get "PE OVERFLOW" error (ERROR 25)...

...if the error occurs occasionally, it may be because you have the limit (PEMAX) set too low. Raise it by 20% and see if the problem is corrected.

...use the VECTOR**STAR** RECORD function to record ICMD when a PE overflow occurs. If ICMD is saturating (equal to ILIM for more than a few milliseconds), you are commanding motion that your motor cannot perform. See hints on motor loading, ILIM, ACC, DEC, and PEMAX below. If the overflow occurs at high speeds and with low ICMD (below ILIM), see the hint about speed problem.

...make sure that the load does not exceed the capability of the motor.

...make sure that ILIM is set high enough.

...if you get the error during acceleration or deceleration, make sure ACC and DEC are not set too high. If they are too high, the commanded profile will exceed the capability of the motor.

...if you get the error during constant speed, verify that the AC line voltage is large enough. Chapter 1 lists the VECTOR**STAR** model numbers. If the voltage you apply to the VECTOR**STAR** is lower than the specified voltage, the motor will not operate properly at high speed.

If you get overspeed errors (ERROR 13)...

...if the error occurs occasionally, it may be because you have the limit (VOSPD) set too low. Raise it by 20% (or as high as 120% of VMAX) and see if the problem is corrected.

...if it happens on acceleration, it may be because your motor is not tuned properly. Is your motor overshooting or ringing? Retuning the motor should correct the problem.

...if it happens when the motor is rotating very slowly so that you are sure that the speed is not near VOSPD, your resolver or R/D converter may have failed. This is simple to confirm. Disable the VECTOR**STAR** and write a program that continuously prints PRD. Rotate the motor slowly by hand and observe PRD to see if it skips several counts (do not be concerned if PRD skips a few counts—look for skips of 50 counts or more). If PRD skips more than 50 counts when the motor is rotating slowly, contact the factory.

If the system works differently on power-up than it does after your program starts running, remember that many switches are reset on power-up. Your program may set a switch that is cleared, or clear one that is set during the initial cycle. After that, the program may operate differently. You may also be setting or clearing switches in your power-up routine that may have the same effect.

11.7 ERROR LOG

The VECTOR**STAR** responds to a variety of conditions, both internal and external, hardware and software, which are grouped in a single broad category: errors. An error indicates that there is a problem somewhere. More serious errors are grouped as faults.

11.7.1 Error Levels

The VECTOR**STAR**'s response to an error depends on the error's severity. There are four levels of severity, listed below in increasing order:

Table 11.3 Error Severity Levels and Actions

1.	Errors that cause warnings.
2.	Errors that cause a program break and stop motion, in addition to Level 1 Actions.
3.	Errors that disable the system and set the FAULT LED, in addition to Level 2 Actions.
4.	Errors that disable almost all VECTOR-**STAR** functions (including communications) and flash the FAULT LED to indicate the error number. These are called firmware errors.

When any error except a firmware error occurs, a message is displayed on the screen. The following items are printed: the error number, the offending entry, and an abbreviated error message. For example, disable the drive and type in a jog:

```
DIS
J 100
```

The VECTOR**STAR** will respond with:

```
ERR 50 'J 100'          VECTORSTAR INHIBITED
```

The error number (50), the offending entry (the whole line), and the error message (you cannot command a jog when the drive is inhibited) are given on one 80-character line. The error message starts at character 40 so that if a 40-character display is used, the error message will not be printed. You can display the line directly, either with the Motion Link Editor (GOTO A LINE NUMBER selection or ^Q^I), or with the VECTOR**STAR** Editor (P command).

Sometimes only an entry is bad and not the whole line. In this case only the bad entry is printed. For example,

```
PROP 2
```

generates:

```
ERR 83 '2' ;BAD OR OUT OF RANGE
```

since PROP is a switch and cannot be set to 2. If the error comes from the program, the line number of the offending entry is also printed. Use the Editor to enter these lines at the top of the user program:

```
11$
PROP 2
B
```

Exit the Editor and type:

```
RUN 11
```

and the response should be:

```
ERR 83 LINE 2 '2'   ;BAD OR OUT OF
                    ;RANGE
```

This message shows that the error occurred on line 2. You can enter the Editor and type:

```
P 2
```

and the line:

```
PROP 2
```

will be displayed.

11.7.2 DEP

If your VECTOR**STAR** prints to a Data Entry Panel (DEP-01) or any other 40-character wide display, the standard error messages will not print properly. The problem is that error messages are based on an 80-character wide display and the DEP-01 is only 40 characters wide. To correct this problem, the VECTOR**STAR** provides the DEP switch, which, when turned on, cuts all error messages down to 40 characters. If your VECTOR**STAR** prints to a DEP-01, type:

```
DEP ON
```

11.7.3 Error History

The VECTOR**STAR** stores the twenty most recent errors in the Error History. To display the entire Error History, type:

```
ERR HIST
```

This causes the Error History to be sent to the terminal, with the most recent error sent first. When the VECTOR**STAR** is powered-up, a "DRIVE POWERED UP" message is inserted into Error History even though this is not an actual error.

To clear the Error History, type:

```
ERR CLR
```

Error History remains intact even through power-down.

11.7.4 Displaying Error Messages

The ERR command can also be used to display an abbreviated description of the error. For example, type:

```
ERR 50
```

The VECTOR**STAR** responds with:

```
ERR 50 VECTORSTAR INHIBITED
```

You may display messages for errors from 1 through 999. If you type in an error number that the VECTOR**STAR** does not recognize, it will respond with:

```
ERROR NOT FOUND
```

A description of all errors is given in Appendix C.

11.7.5 Firmware Errors

Firmware errors are an indication of a serious problem with the VECTOR**STAR**. These errors stop communications, disable the drive, and flash the FAULT LED. The FAULT LED flashes several times, then turns off and pauses. The number of flashes represents the error number. These error numbers range from 2 to 9. See Appendix C for information on these errors. Contact the factory should one of these errors occur.

CHAPTER 12
HIGH POWER

12.1 INTRODUCTION

The VECTOR**STAR** Series of drives consists of two families: the VSA modular units and the VSL Series.

The VSA series features 230-volt units that are available to 28 KW (85 Amps continuous). The VSA units require an external PA series power supply of the appropriate current rating. Axis drives are also powered from this same power supply. All units use a resolver as a feedback device. The PA series power supply can also be a stand alone, line regenerative supply in an 110 Amp rating. The PAR110 model adds line regenerative capability to the PA110 unit.

The VSL/P series features integral power supplies and is available to 90 KW. The VSP can be applied on 460 VAC lines. Line regeneration is an option with the VSL/P series.

Programmable control is identical in both the VSA and VSL/P series. The "Motion Link" communications language is used and configured to match the motor and application with the same TL and Variables parameters. The power interface is different for these 2 product lines due to the difference in power levels. However, all VSA power levels use the same power interface, and all VSL/P power levels use a common power interface design. Mechanical package, power elements, cooling, and scaling set the power level for each model of the VSL/P drive, but all control cards are identical between units.

This modular approach simplifies inventory, build, testing, and understanding of the Kollmorgen high power drives. The customer can reduce the inventory needed for maintaining the drives and can transfer knowledge of programming and maintenance from one size drive to another, because operational theory is identical.

12.2 VSA DRIVES

The VSA Drives are composed of three distinct control sections.

1. The MC3 Control Card (A-95078) contains the micro and its associated peripheral components, the R/D conversion circuitry, the A/D converter circuitry, I/O circuits and several housekeeping and monitor circuits. This card can accept either analog command or RS232/485 serial inputs.

2. The VECTORSTAR-MIBD (A-93285) card contains analog current loops, opto isolation to isolate the control section from the power section and the six independent gate drive circuits. Overcurrent protection is also part of this card.

3. The IGBT Power Stage, which boosts the control inputs to the necessary levels to drive an induction or PM motor. IGBT power ratings are determined by the KW rating of the VSA drive. Hall effect current sensors provide a continuous monitor of motor phase currents.

12.2.1 Power Supply

Logic and main DC bus voltages are provided by the external PA series power supply on all VSA drives.

12.3 VSL/P SERIES DRIVES

The VSL series drives are composed of six distinct sections.

1. The Power supply and LEM (A-95125) contains the LEM motor current sensors and the logic power supply. This card can be configured for 230 VAC or 460 VAC operation and provides transformer isolation from the AC line. The outputs are +/-15 VDC, +5 VDC, and +14 VDC.

2. The Interface card (A-95140) provides connection from the MC3 control card (described above) and the Gate Drive card (A-95117). This card also receives the inputs from the resolver and sends these signals to the MC3 card for conversion to a digital word. A customer output section provides relay isolation for process interface. Protection circuitry and monitor functions interface to the digital section on the MC3 card, through the Interface card.

3. Gate Drive Card (A-95117) contains the current loops, triangle wave generator for the PWM control, dead time circuits, and six isolated power drivers that control the IGBT gates. The isolated gate drives each have their own power supplies and isolated inputs to insulate the power section from the control section. This card provides overcurrent, overvolts, undervolts, and over temperature protection circuitry for the drive. Remote enable and motor brake sensing is provided as a means of interface to customer processes that require these functions. The Gate Drive card has been designed to accept drive signals from an external control systems and provides the correct gating signals to the power devices. Current sensing is done by externally mounted LEM modules.

All drives, regardless of rating, use the same "Motion Link," variables scheme, and TL functions, so it is easy for the user to interface the drives to specific processes and machine tools. Since cards are common, understanding of these systems is easier and parts inventory is reduced. Internal wiring for the Integral Power Supply units is identical.

Since this is a fully programmable system, no additional or option cards are needed to provide other functions.

12.4 INSTRUCTION FOR THE LINE REGENERATION UNIT

12.4.1 General Description

The Line Regeneration Unit is a regulated, three-phase AC/DC power supply. Its functions are line regeneration and power factor correction. The main application is for servo drives.

The unit is actually a bi-directional power converter. In rectifier mode, the unit converts the three-phase line voltage to a DC voltage. This is the same function as a full bridge rectifier but with a sinusoidal current taken from the line and a DC bus voltage that is regulated higher than a bridge rectifier can achieve.

The regenerative mode occurs when the DC bus voltage is higher than the rectifier mode. A good example is when a motor acts as a generator. The unit sends power back to the three-phase line input in this mode.

The unit senses the phase of the input line voltage and tries to make the power factor as close as possible to unity regardless of the load. If the line current is within the rating of the unit for both directions, the DC bus voltage will be a constant value.

12.4.2 Performance

System rating:	100 A rms continuous and 200 A rms peak line current for both rectifier and line regeneration modes.
DC bus regulation:	345 volts +/- 5% on 240 VAC Input 690 volts +/- 5% on 480 VAC Input
Drive Protection:	Overvoltage for DC bus and undervoltage for line, overcurrent, phase loss, and over temperature. A relay will be dropped out if any of the above faults occurs.
THD (harmonic distortion)	4.5% - 6.5 with a constant load

12.4.3 Fault Relay

Before operating the unit, connect the relay of the line regen (pins 1 and 2 of J6) to the drive REMOTE input. In this way, the drive will be disabled when there is a fault from the line regen.

High voltage will be on DC bus at all times unless the three-phase line voltage is removed.

12.4.4 Operating the Line Regen

In normal operation, no adjustment is necessary and the unit is always ready to operate without any special settings. The LED will indicate the unit status/error message and a reset button can be used to reset the unit.

The three-phase input voltage must be higher than 195 V rms (line to line) to operate the unit. After turning on the main power to the Line Regen Unit, the unit will be in a soft-start procedure. The soft-start procedure ends when the contactor closes.

High voltage will be on DC bus immediately after main power is applied.

About 1.5 seconds after the contactor is closed, the unit will start to regulate the DC bus voltage and enter the normal working state. An “E” (Enabled) will be shown on the LED.

12.4.4.1 Error Message on the LED and Possible Reasons

o Over voltage (Bus voltage cannot be held on regen; you may have tried to regen too hard).

i Over current or over temperature.

L Phase loss of line voltage.

12.4.4.2 RS-232 Serial Port

Normally, communication with the Line Regen Unit is not recommended. The statement here is only for troubleshooting. The Line Regen Unit is programmable and has an RS-232 serial port for the user to communicate with the unit. RS-232 communication allows the compensation to be adjusted if required. Also, the system parameters and user program may be checked and modified. The RS-232 can also be used to monitor the system working states. To do this, you must use the BDS5 Motion Link software and a PC.

Compensation parameters:

VCMD: Voltage command; should be 345 on 240 VAC systems or 690 on 480 VAC systems.

KP: Proportional gain for the DC bus voltage regulation. Set to 10-30.

KVI: Integrating gain for the DC bus voltage regulation. Set to 1-4.

KPROP: DC bus over voltage. Set to 900-950 (400-430 volts). If the DC bus voltage is greater than this, the unit will shut down automatically.

Line Regen Unit hardware overvoltage protection trips around 420 volts DC.

ILIM: Current limit. In units of percentage of rated peak current, with a range of 1-100.

The following parameters show the unit working states:

V DC bus voltage, actual volt = V/2.3 for 230 V system.

VP Line peak voltage.
Actual line rms volt = VP + 10 for 230 V system.

ICMD Current command, in units of percentage of rated peak current.

IMON Actual rms line current, same units as ICMD.

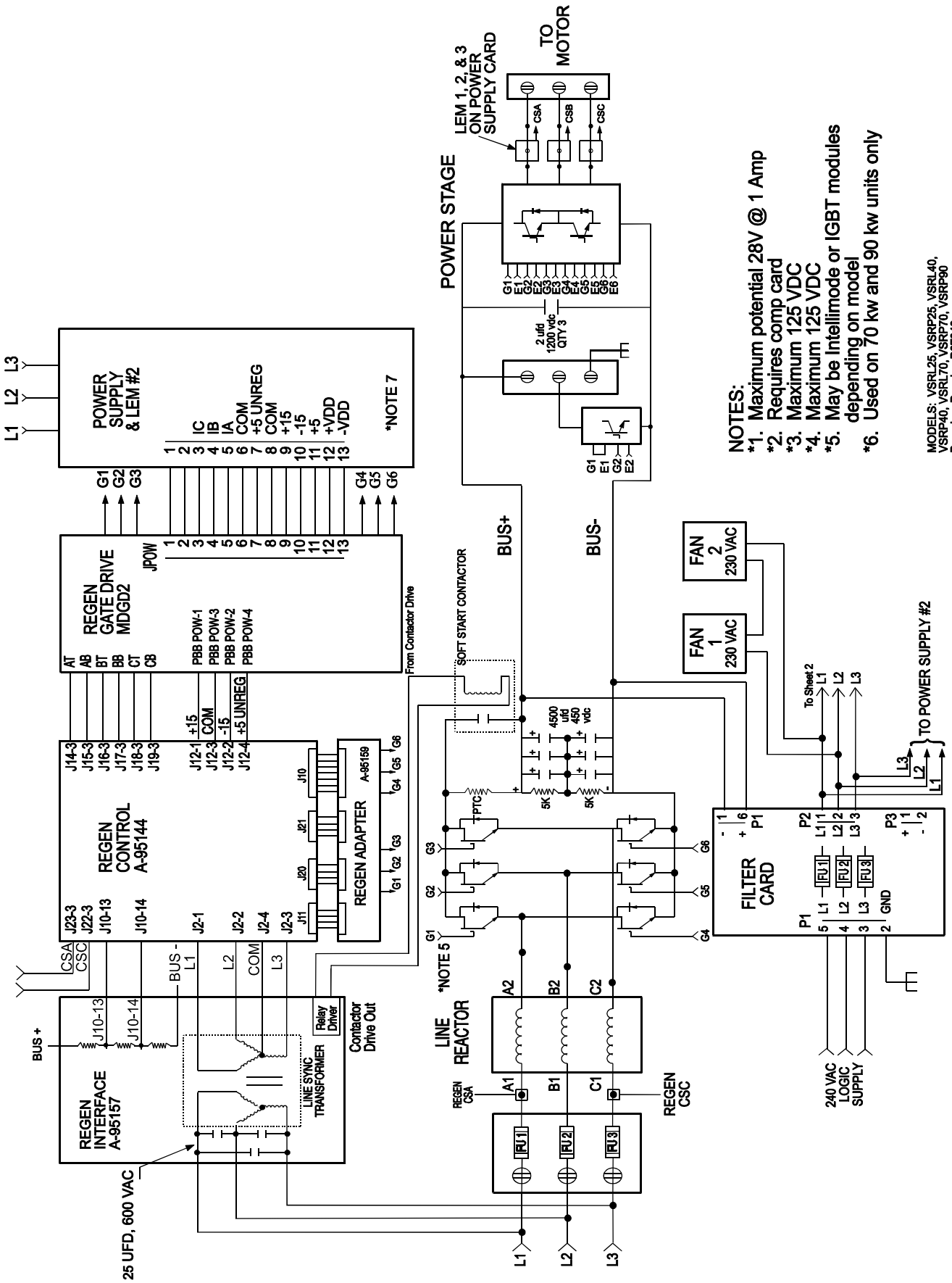


Figure 12.1 Line Regen System Diagram, Sheet 1

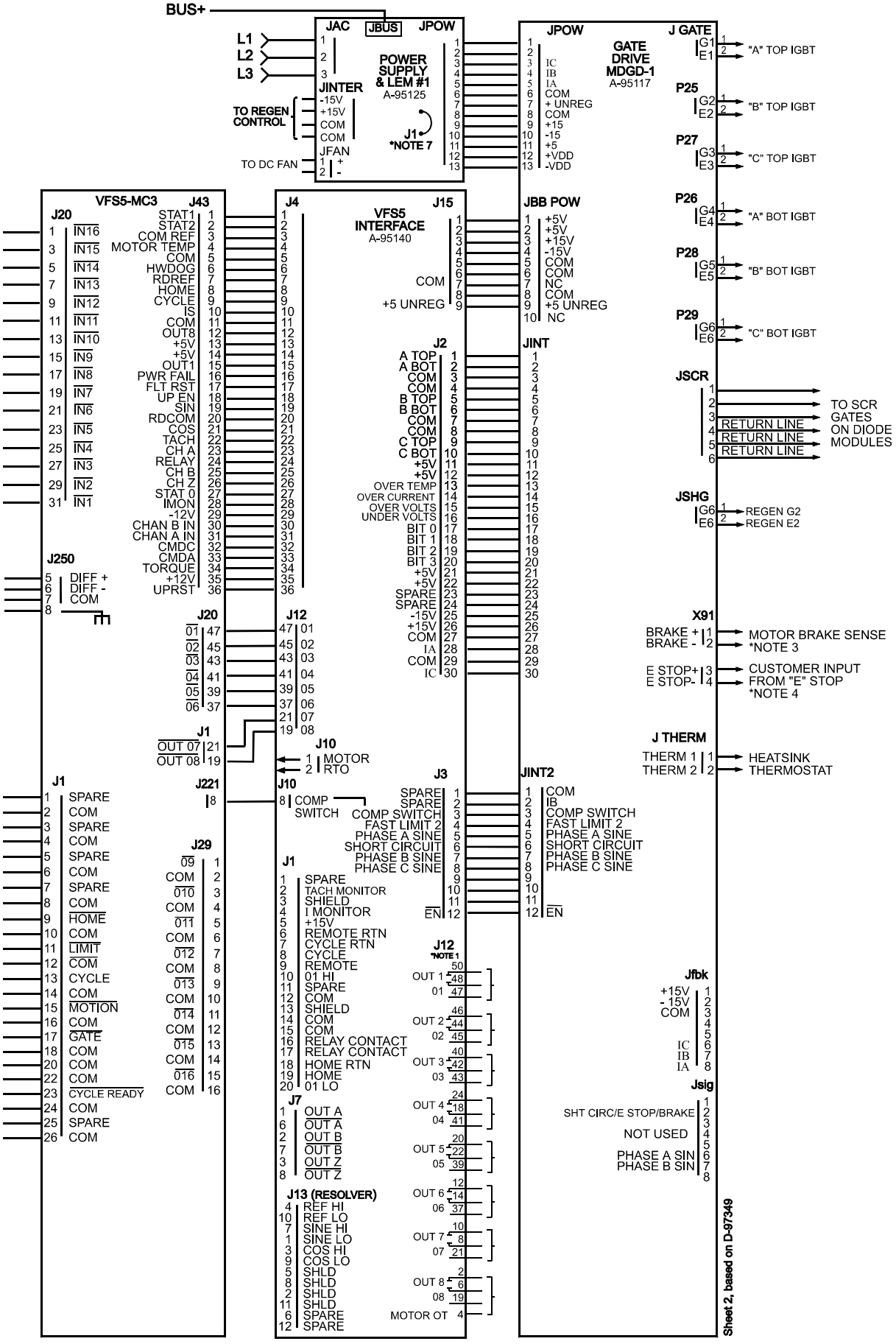


Figure 12.2 Line Regen System Diagram, Sheet 2

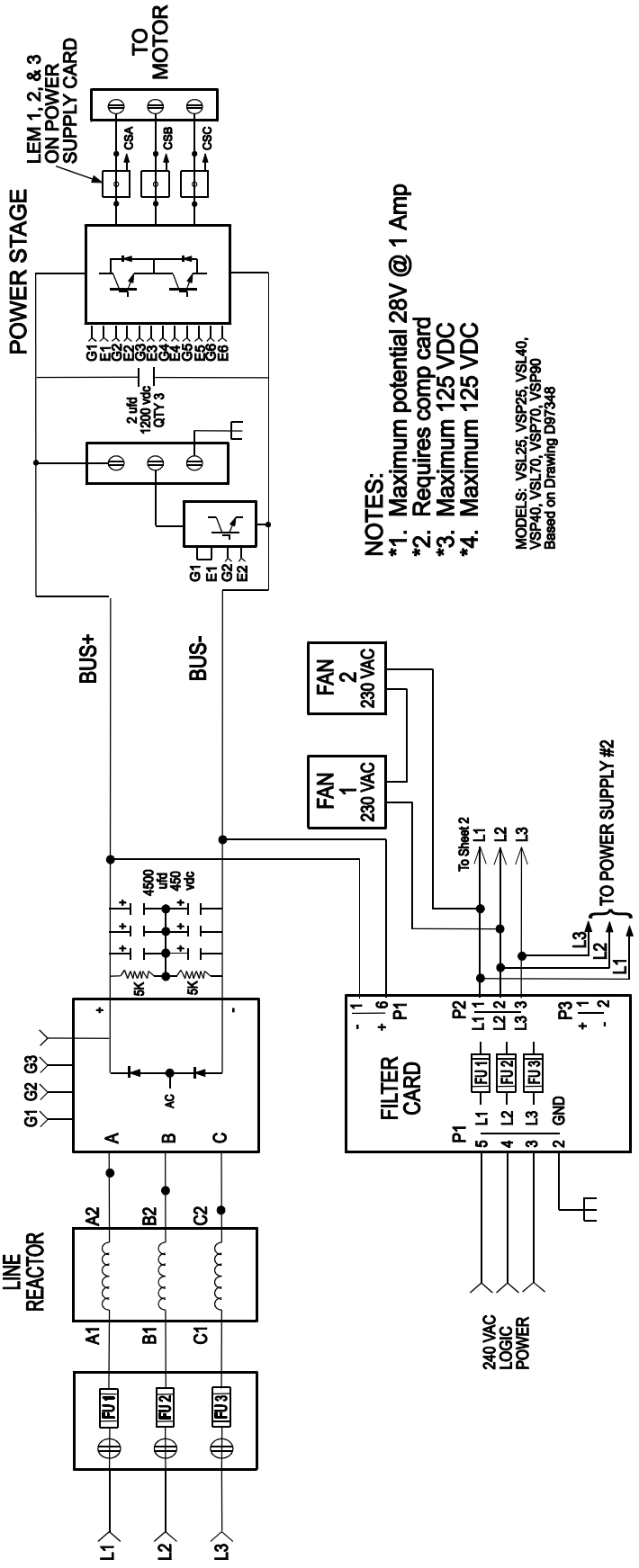


Figure 12.3 VSL, VSP Resistive Regen System Diagram, Sheet 1

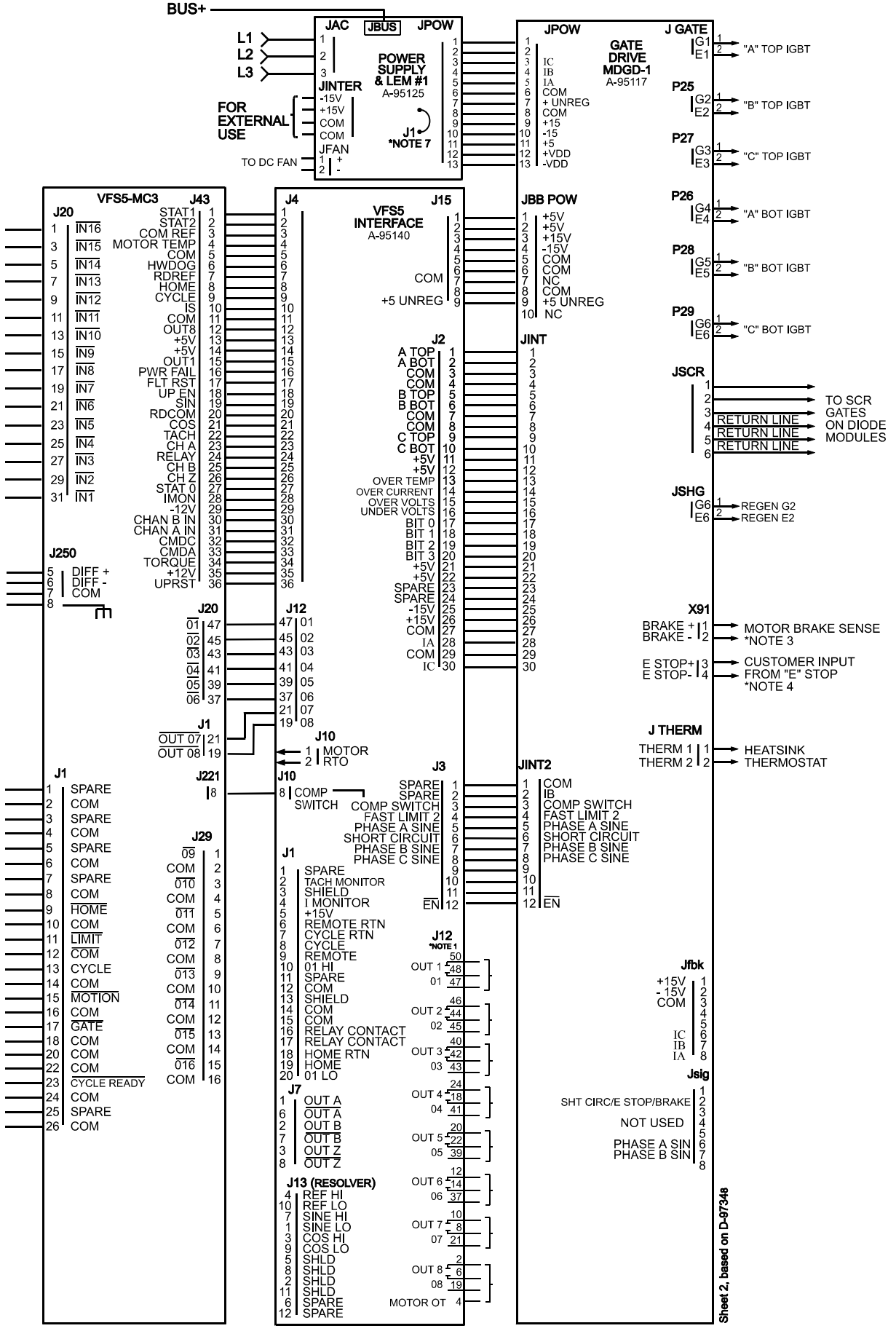


Figure 12.4 VSL, VSP Resistive Regen System Diagram, Sheet 2

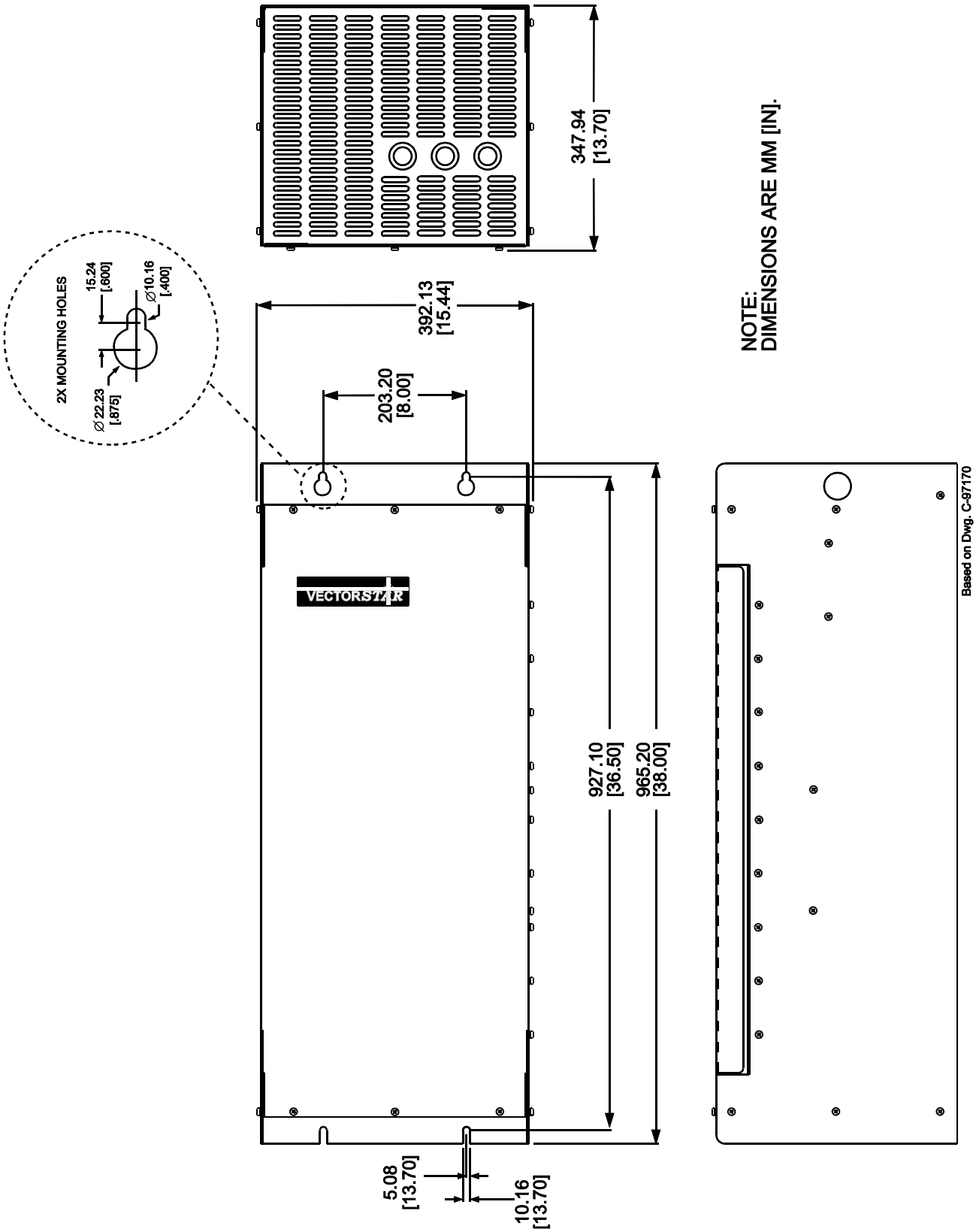


Figure 12.5 Outline and Dimensions VSL70

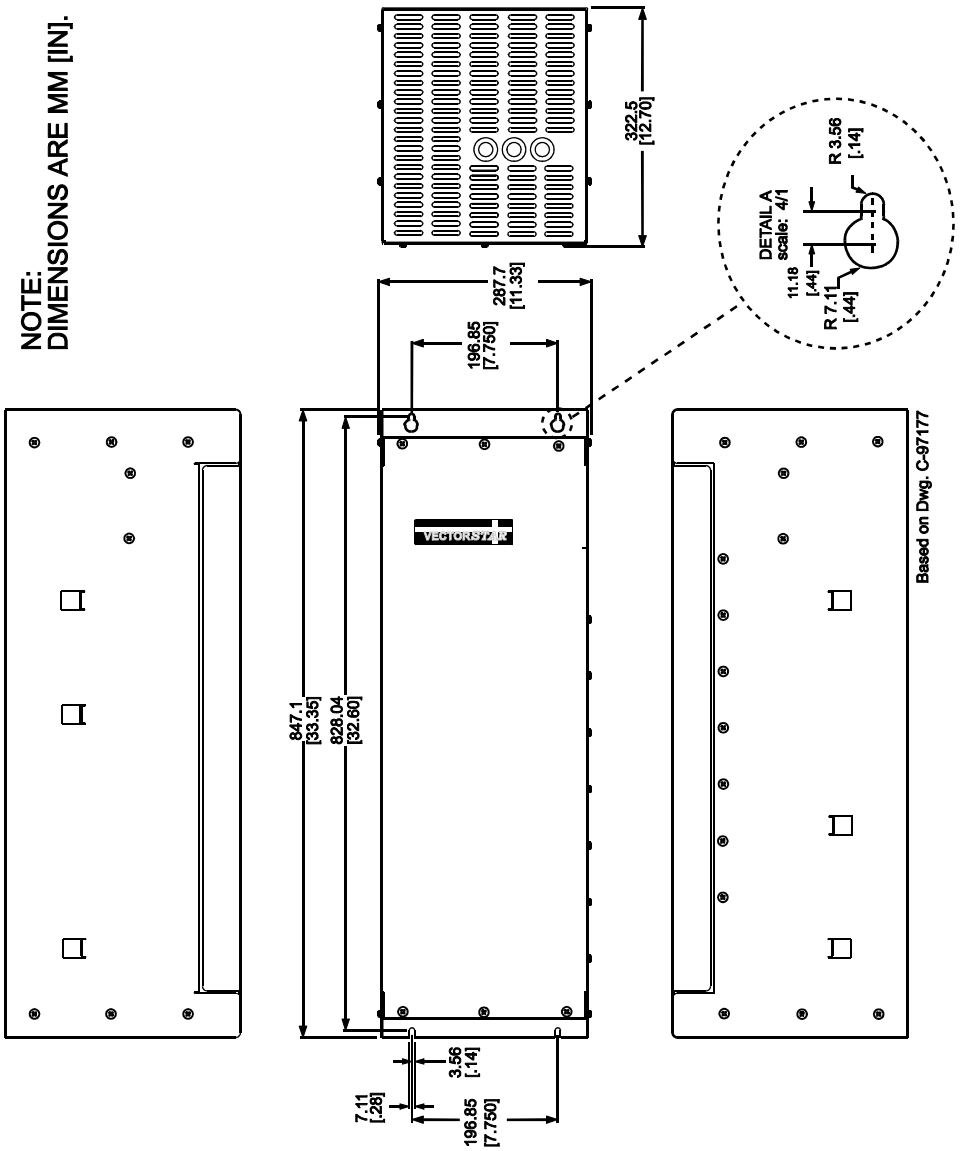


Figure 12.6 Outline and Dimensions VSL40

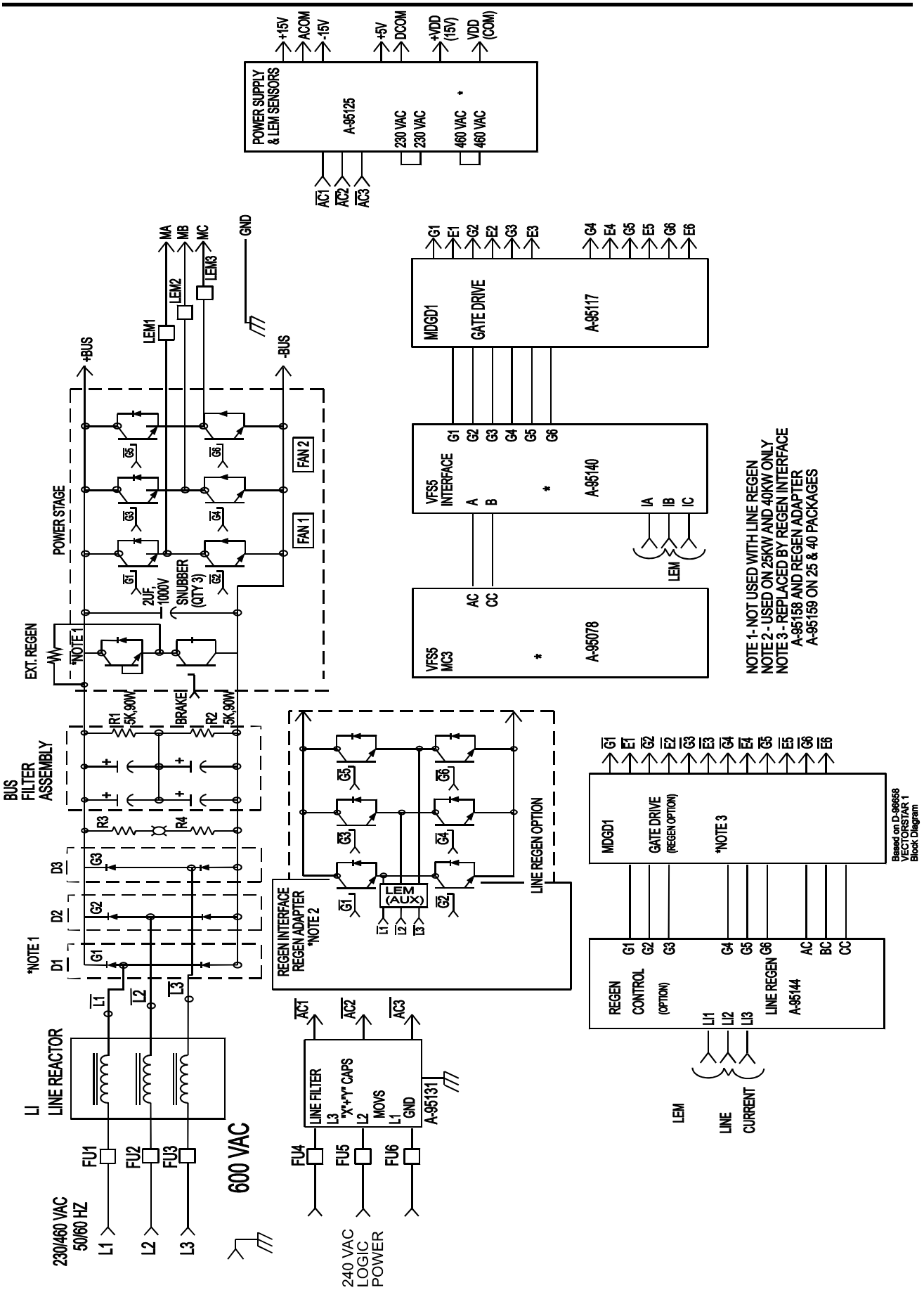


Figure 12.7 High Power VS(R)L, VS(R)P Block Diagram

MODEL	FUSE BLOCK	FUSES FU1, FU2, FU3	REACTOR L1	DIODE/THYRISTOR MODULE D1,D2,D3	REGEN UNIT	BUS CAPS ASSEMBLY
VSL25-0012	T60200-IC P-97342-001	JLLN, 200A, 300V P-97345-009	P-97335-003 KLR80BTB	115A, 1200V A-84990-012	CM300DY-12H A-93715-300	CAPACITOR CARD A-96432
VSP25-0012	T60100-3C P-97342-002	JLLS, 100A, 600V P-97345-021	P-97335-001 KLR45BTB	106A, 1200V A-93711-012	MG100Q2YS11 P-97330-100	CAPACITOR CARD A-96432
VSRL25-012	T60200-IC P-97342-001	JLLN, 200A, 300V P-97345-009	P-97335-003 KLR80BTB		PM200CSA260 A-97051	CAPACITOR CARD A-96432
VSRP25-0012	T60100-3C P-97342-002	JLLS, 100A, 600V P-97345-021	P-97335-001 KLR45BTB		PM150CSA120 P-97337-001	CAPACITOR CARD A-96432
VSL40-0012	T60200-IC P-97342-001	JLLN, 200A, 300V 9-97345-009	P-97335-004 KLR110BCB	150A, 800V A-96354-008	CM300DY-12H A-93715-300	CAPACITOR CARD A-96432
VSP40-0012	T60100-3C P-97342-002	JLLS, 100A, 600V P-97345-021	P-97335-002 KLR55BTB	106A, 1200V A-93711-012	MG200Q2YS11 P-97330	CAPACITOR CARD A-96432
VSRL40-0012	T60200-IC P-97342-001	JLLN, 200A, 300V P-97345-009	P-97335-004 KLR110BCB		PM300CSA660 P-97338	CAPACITOR CARD A-96432
VSRP40-0012	T60100-3C P-97342-002	JLLS, 100A, 600V D-97345-021	P-97335-002 KLR55BTB		PM150CSA120 P-97337	CAPACITOR CARD A-96432
VSP70-0012	T60200-IC P-97342-002	JLLS, 200A, 600V P-97345-026	KLR110BCB P-97345-026	150A 1200V A-96354-012	MG300Q2YS11 P-97731-001 QTY1	P-97334-001 QTY 4
VSRP70-0012	T60200-IC P-97342-002	JLLS, 200A, 600V P-97345-026	KLR110BCB 9-97335-004		MG300Q26S11 P97731-001 QTY 3	P-97334-001 QTY 4
VSP90-0012	BH-1111 P-97339-002	JLLS, 250A, 600V P-97345-028	KLR130BCB P-97335-005	150A, 1200V A-96354-012	MG400Q1US41 P-97332-001 QTY 2	P-97334-001 QTY 6
VSRP90-0012	BH-1111 P-97339-002	JLLS, 250A, 600V P-97345-028	KLR130BCB P-97335-005		MG400Q1US41 P-97332-001 QTY 6	P-97334-001 QTY 6

Figure 12.8 High Power VS(R)L, VS(R)P System Matrix Component LIstings

MODEL	HEATSINK	IGBT POWER STAGE	REGEN CONTROL	AC VOLTS	S1 CONT.	S2 30 MIN	Peak 2 SEC
VSL25-0012	C-96369	CM300DY-12H A-93715-300		230 VAC RESISTIVE REGEN	80 A	104 A	160 A
VSP25-0012	C-96369	MG100Q2YS11 P-97330-100		460 VAC RESISTIVE REGEN	40 A	52 A	80 A
VSRL25-012	C-96369	CM300DY-12H A-93715-300	A-95144 CONTROL A-95158 INTERFACE A-95159 ADAPTER	230 VAC LINE REGEN	80 A	104A	160A
VSRP25-0012	C-96369	MG100Q2YS11	A-95144 CONTROL A-95158 INTERFACE A-95159 ADAPTER	460 VAC LINE REGEN	40 A	52 A	80 A
VSL40-0012	C-96369	CM300DY-12H A-93715-300		230 VAC RESISTIVE REGEN	100 A	133 A	200 A
VSP40-0012	C-96369	MG200Q2YS11 P-97330		460 VAC RESISTIVE REGEN	56 A	73 A	112 A
VSRL40-0012	C-96369	CM300DY-12H A-93715-300	A-95144 CONTROL A-95158 INTERFACE A-95159 ADAPTER	230 VAC LINE REGEN	100 A	133 A	200 A
VSRP40-0012	C-96369	MG200Q2YS11 P-97330	A-95144 CONTROL A-95158 INTERFACE A-95159 ADAPTER	460 VAC LINE REGEN	56 A	73 A	112 A
VSP70-0012	D-96654	MG300Q2YS11 P-97331-001		460 VAC RESISTIVE REGEN	105 A	137 A	210 A
VSRP70-0012	D-96654	MG300Q2YS11 P-97331-001 QTY 3	A-95144 CONTROL A-95117 GATE DRIVE	460 VAC LINE REGEN	105 A	137 A	210 A
VSP90-0012	D-96654	MG400Q1US41 P-97332-001 QTY 6		460 VAC RESISTIVE REGEN	130 A	170 A	260 A
VSRP90-0012	D-96654	MG400Q1US41 P-97332-001 QTY 6	A-95144 CONTROL A-95117 GATE DRIVE	460 VAC LINE REGEN	130 A	170 A	260 A

Figure 12.8 (Continued) High Power VS(R)L, VS(R)P System Matrix Component LIstings

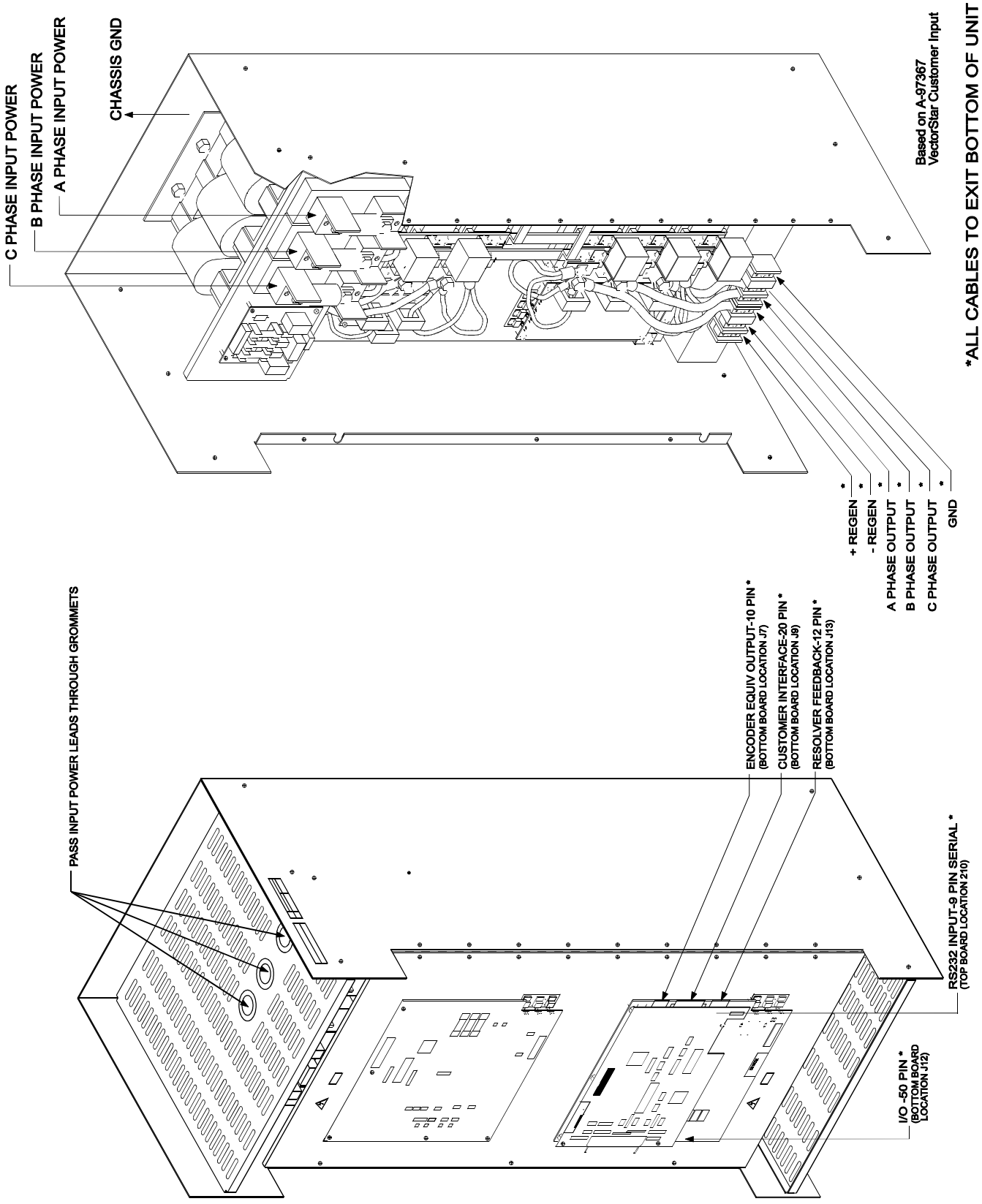


Figure 12.9 VS(R)L, VS(R)P Connector Locations

APPENDIX A
WARRANTY INFORMATION

Kollmorgen Corporation warrants that equipment delivered by it to the Purchaser will be of the kind and quality described in the sales agreement and/or catalog and that the equipment will be free of defects in design, workmanship, and material.

The terms and conditions of this Warranty are provided with the product at the time of shipping or in advance upon request.

The items described in this manual are offered for sale at prices to be established by Kollmorgen and its authorized dealers.

APPENDIX B
DRAWINGS

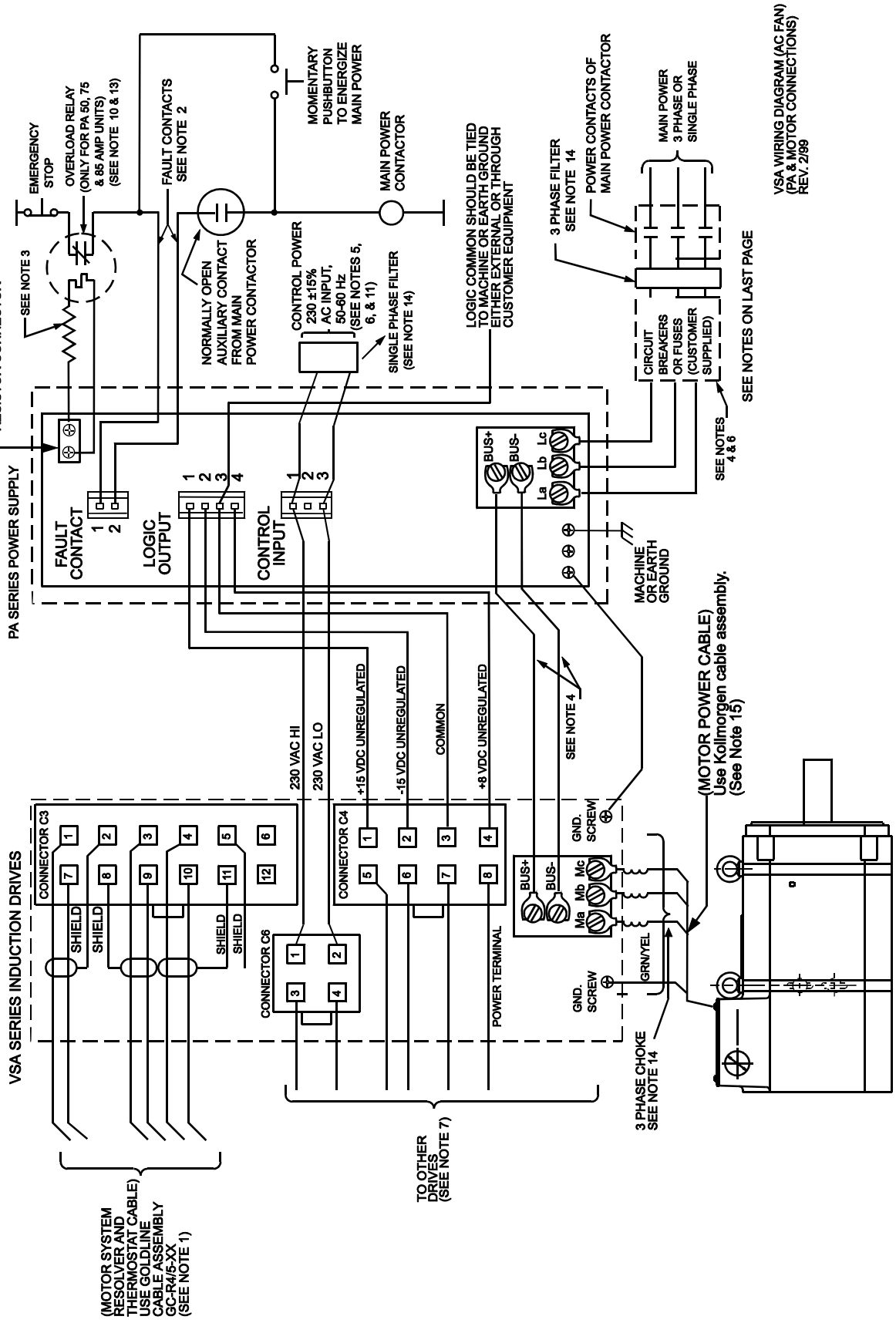


Figure B.1 PA and Motor Connections (AC Fan)

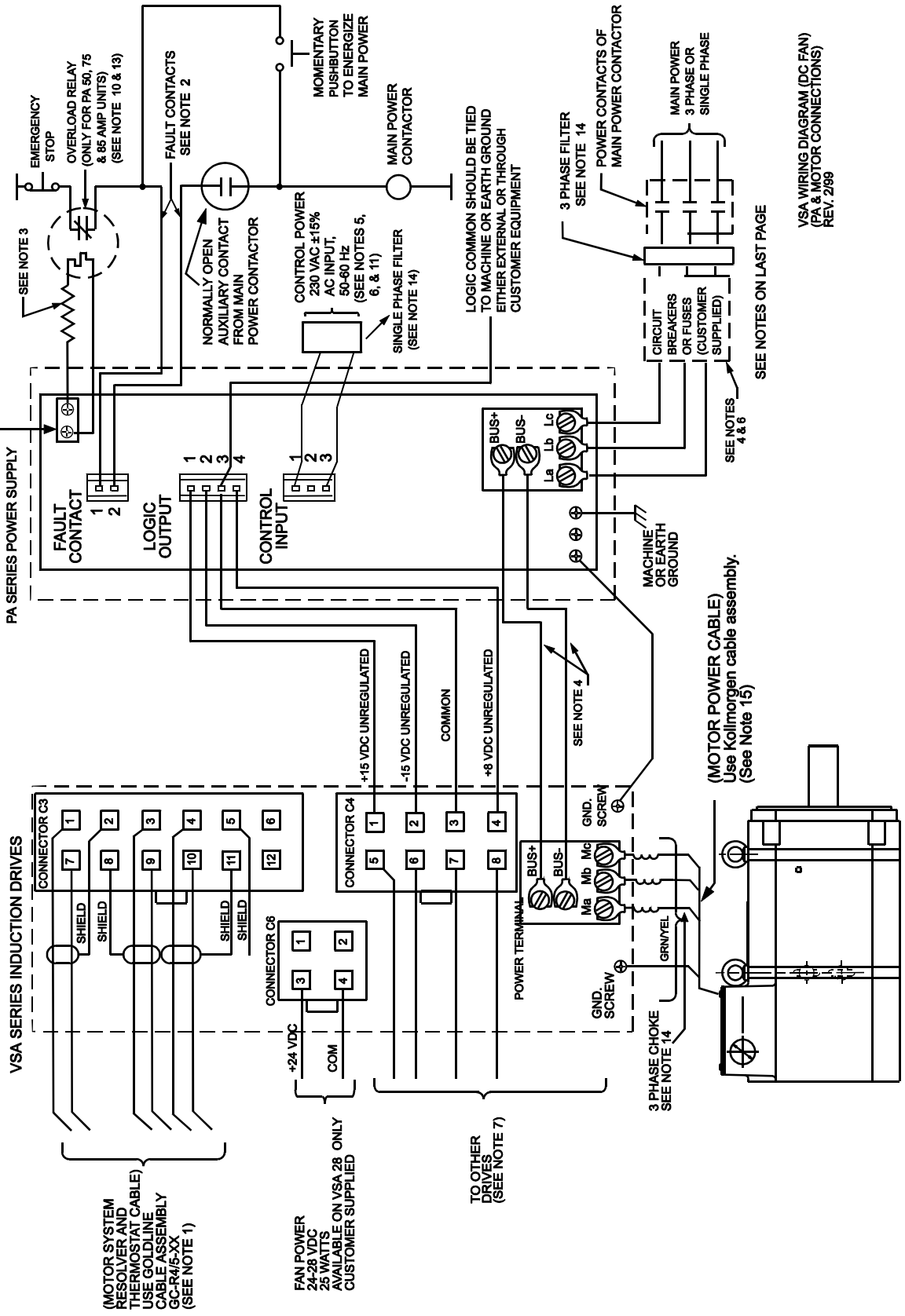


Figure B.2 PA and Motor Connections (DC Fan)

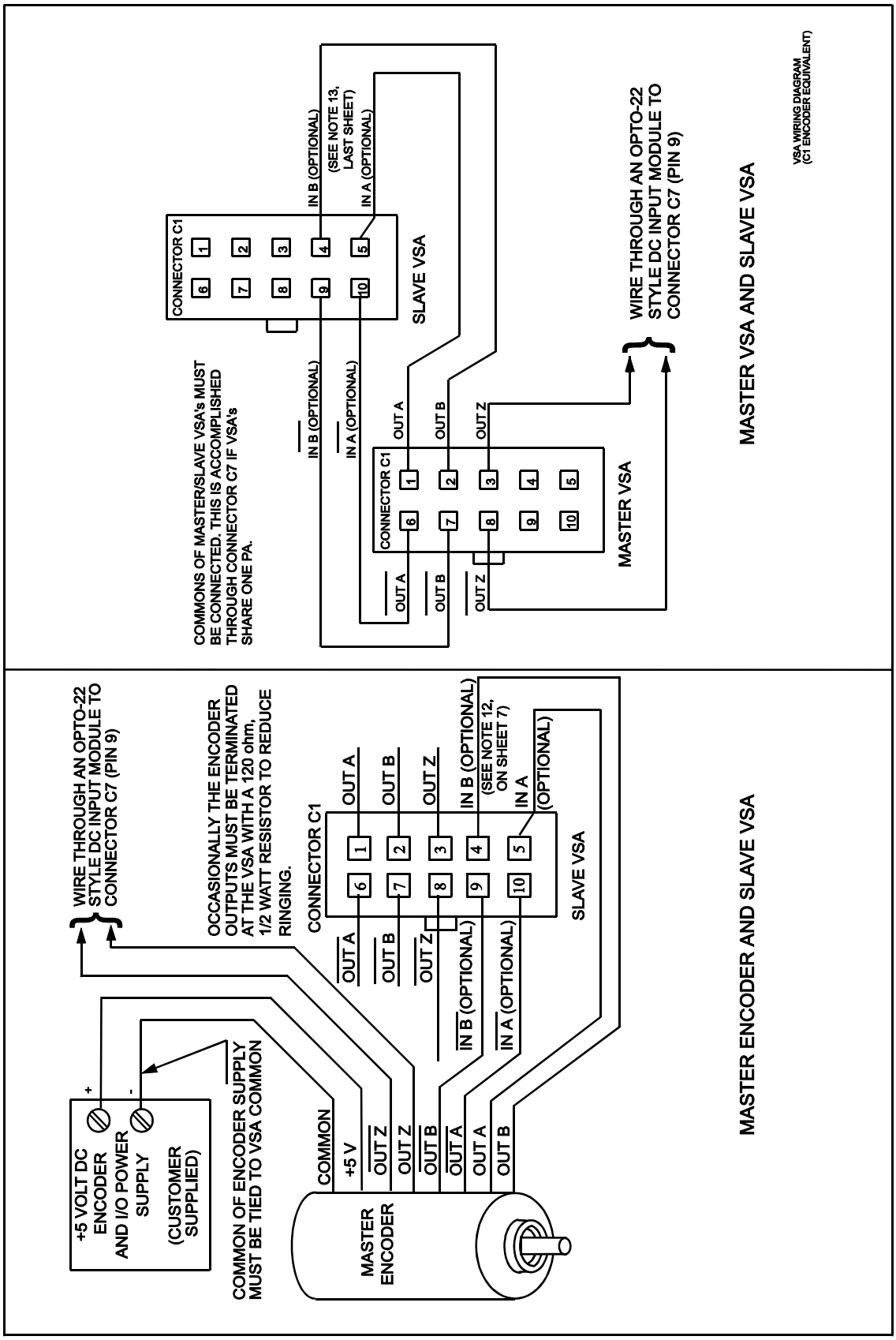


Figure B.3 C1 Encoder Equivalent

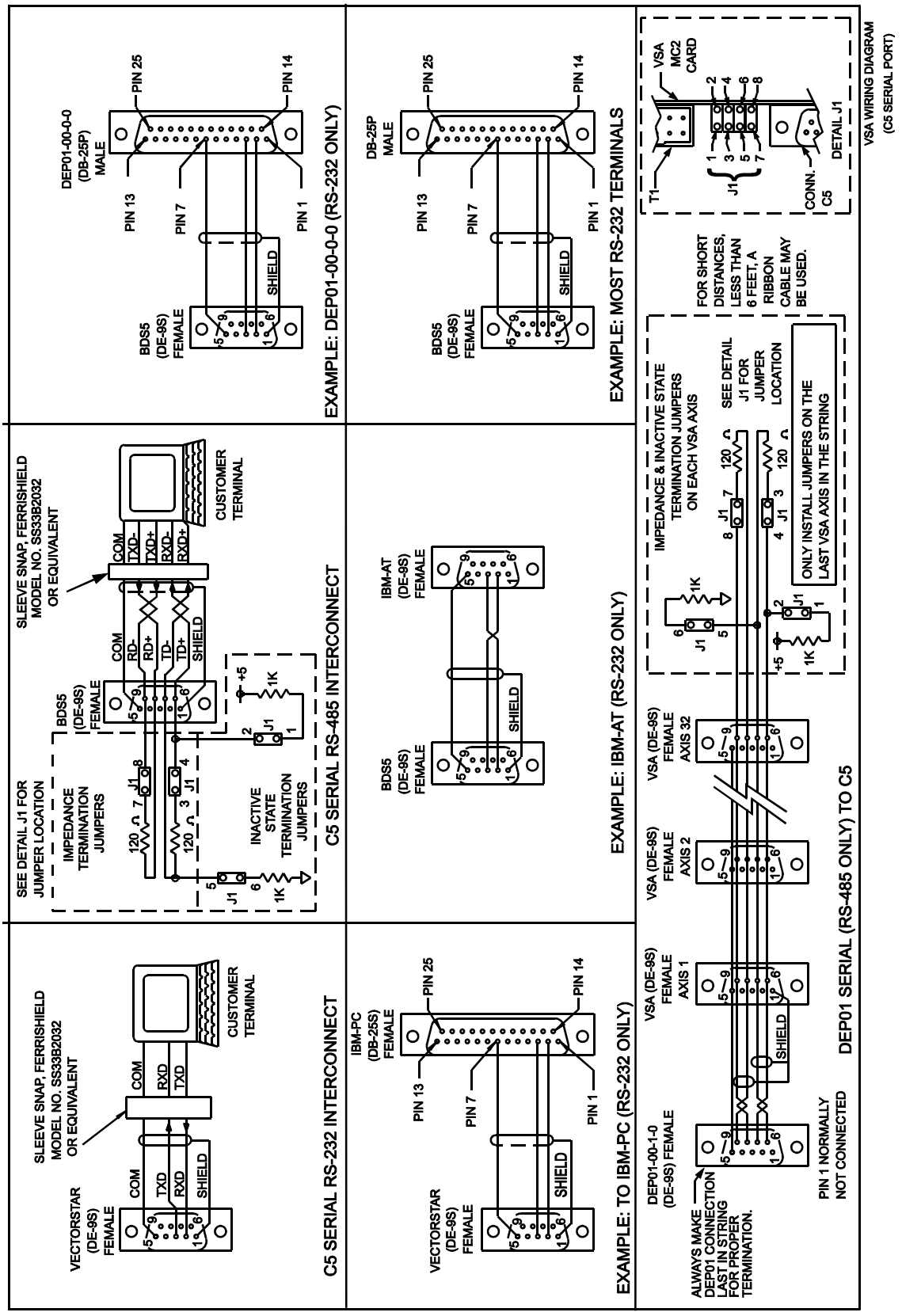


Figure B.4 C5 Serial Port

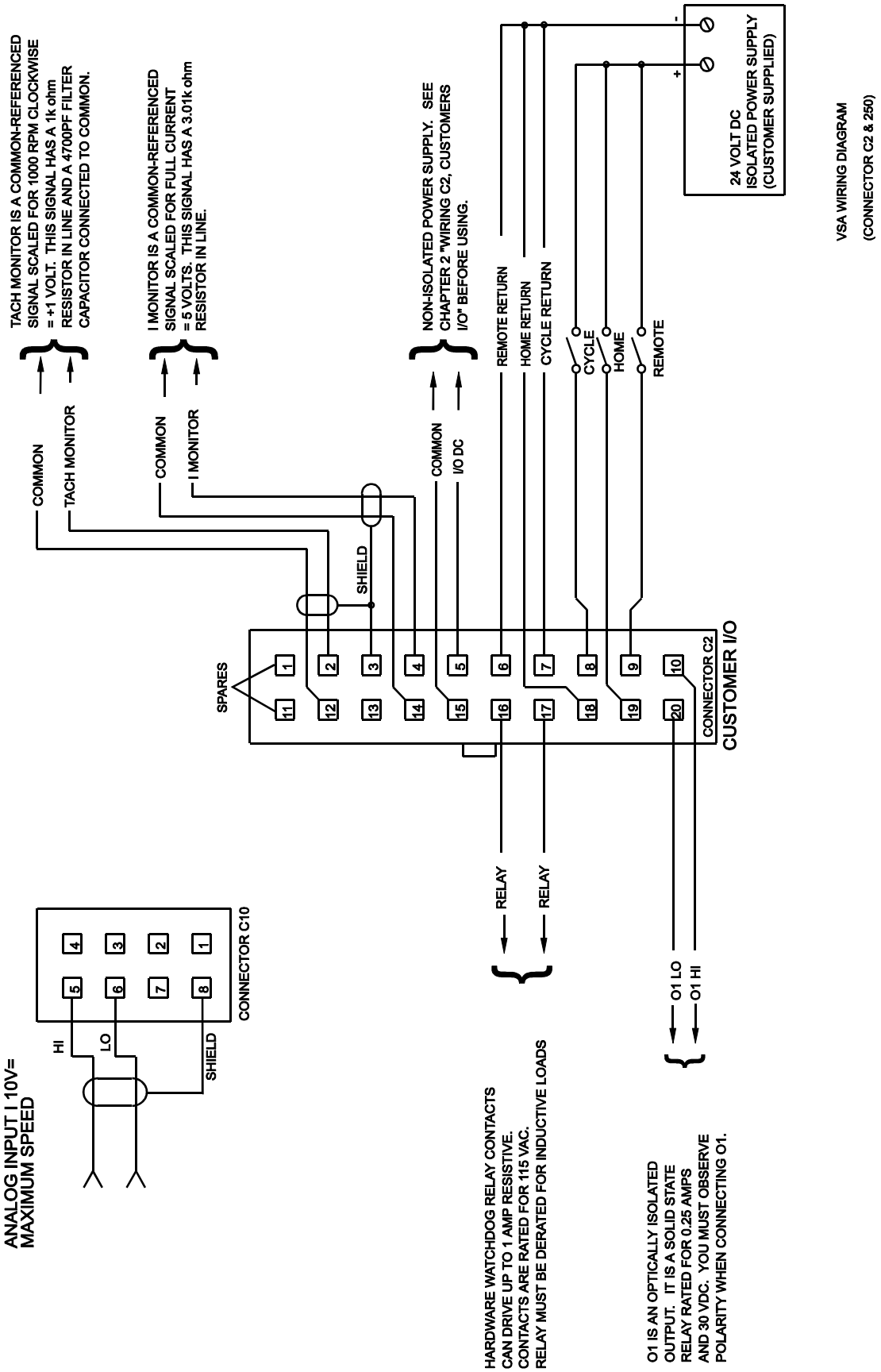


Figure B.5 Connector C2 and 250

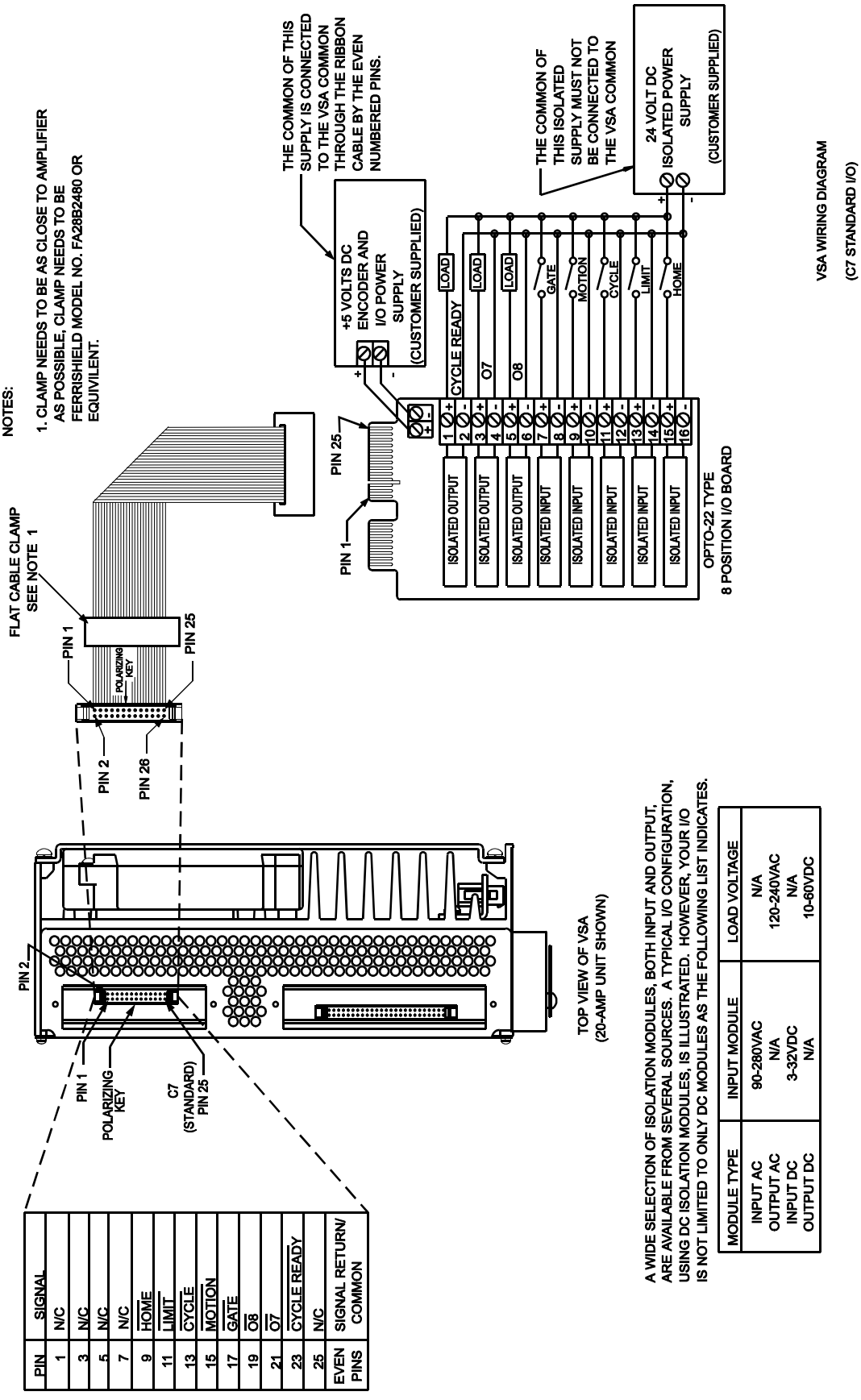


Figure B.6 C7 Standard I/O

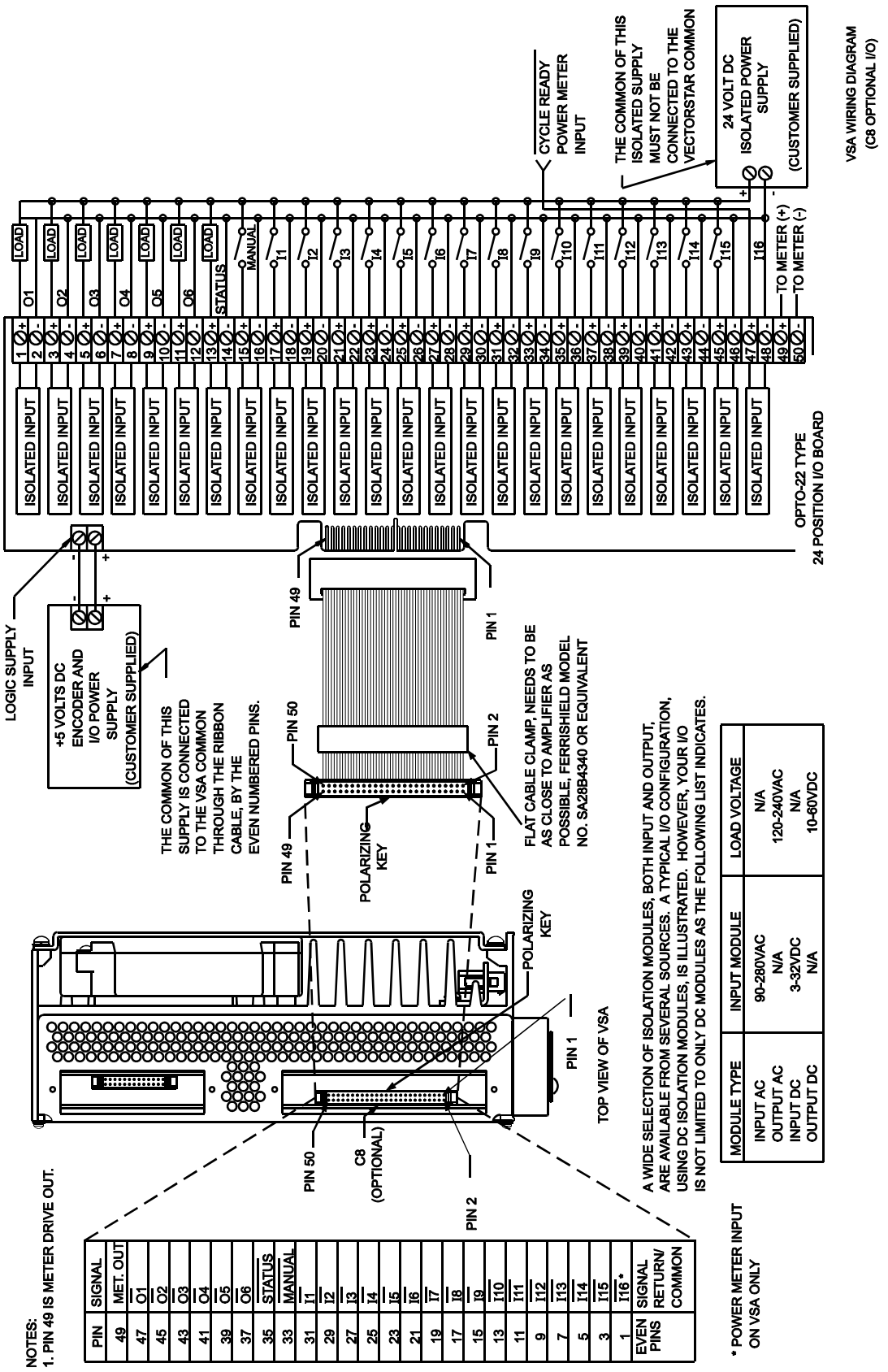


Figure B.7 C8 Optional I/O

NOTES TO ACCOMPANY FIGURES B.1-B.7

(ALL WIRES TO BE COPPER WITH MIN. TEMP. RATING OF 60°C)

1. **WARNING:** The motor thermostat automatically resets when the motor cools. The customer is responsible for latching this signal to inhibit operation after a motor thermostat fault. connect thermostat using twisted pair wire.

2. **CAUTION:** The PA fault contacts (rated 115 VAC 1 AMP) must be wired in series with the overload relay as shown on Figure B.1. On the 50, 75, and 85 AMP PA, this contact closes on application of control power and after DC bus is active.

3. **CAUTION:** Resistor is connected to high voltage; ensure sufficient electrical clearance mounting. Resistor may become very hot during operation. Do not mount near materials that are flammable or damaged by heat. Ventilation may be required. See wiring drawing for specific regen resistor kit. Each kit has different series/ parallel resistor connection to obtain specific resistance and power rating.

4. Wire sizes, breakers, and fuses for PA:
   PA-50 has a maximum main power input current of 50 AMPS RMS
   PA-75 has a maximum main power input current of 75 AMPS RMS
   PA-85 has a maximum main power input current of 85 AMPS RMS

   The actual application may require less current/ Use 600 VAC insulated wire and refer to local electrical codes for proper wire size for the currents listed above. Fuses for main power should be U.L. rated time delay type, such as Bus FRN-R Series.

   The power bus between a PA and VSA should use the following wire gauge with 600 VAC insulation:
   PA-50, bus bars supplied with unit, or 8 AWG (or larger) wire
   PA-75, bus bars supplied with unit, or 8 AWG (or larger) wire

5. All signals and control wires to be 22-18 AWG wire. The crimp terminals for 22-18 AWG wire are supplied for use with VSA Connectors C1, C2, C3, C4, C6, and PA connectors C6 and C7 and C2. For 16 AWG wire use Molex #39-00-0078 terminals.

6. All AC lines should be twisted cables.

7. The total number of axes allowed per PA depends on the PA model and combination of VSA's and/or SR's:
   PA-50: A maximum of 4 VSA's or 3 SR's.
   PA-75: A maximum of 4 VSA's or 3 SR's.
   Axis expansion on the PA 50, 75, and 85 AMP units are also limited to a maximum of 4 VSA's or 3 SR's on either side of the PA.

8. The VSA is configured at the factory for either RS-232 or RS-485.

9. XX in the cable number stands for cable length in meters. Cable length is available from 3 to 75 meters in increments of 3 meters.

10. A thermal overload relay is supplied in the regen resistor kit for the 50, 75, and 85 AMP PA's. The thermal overload relay, included in the kit, was sized for your resistance and power rating. the output contacts of the relay must be wired to drop power to the main power contactor in a fault condition, as shown in Figure B.1.

11. Do not wire control power (PA connector C8) through the main power contactor. This is so that control power won't be removed if PA fault contacts open (this would turn off any fault LEDs).

12. If the VSA uses the optional analog input card (VSA-OPT1), the optional encoder inputs in Connector C1 are used.

13. Recommended torques for connection to terminal blocks and ground.
    A. VSA-30 to 55 AMP B ground screw.
    B. PA-50 to 75 AMP:
       Max torque 20 in/lb main power, bus connection and ground stud
       Max torque 12 in/lb external regen connection

    For grounding to machine or earth ground, a screw lug should be attached to ground screw or stud. Recommended torque of 12 in/lb for ground screws and 20 in/lb for ground studs. May also refer to National Electric Code (NEC) or UL Standard 486B for recommended torques.

    Thermal overload protection is not provided internal to amplifier and must be provided externally. Refer to National Electrical Code for proper sizing of overload protection.

14. Filter must be provided external to the equipment to meet CE requirements. Contact factory for details.

15. (Motor Cable Power) Use Kollmorgen Cable Assembly. Example: Model No. for B-20X motor is GC-M2-4/5-xx (See Note 9). Thermal overload protection is not provided internal to amplifier and must be provided externally. Refer to National Electric Code for proper sizing or overload protection. It is recommended that motor wiring be shielded cable. Use proper size wire. Use copper wire and 60°C or 75°C table from NEC for wire size.

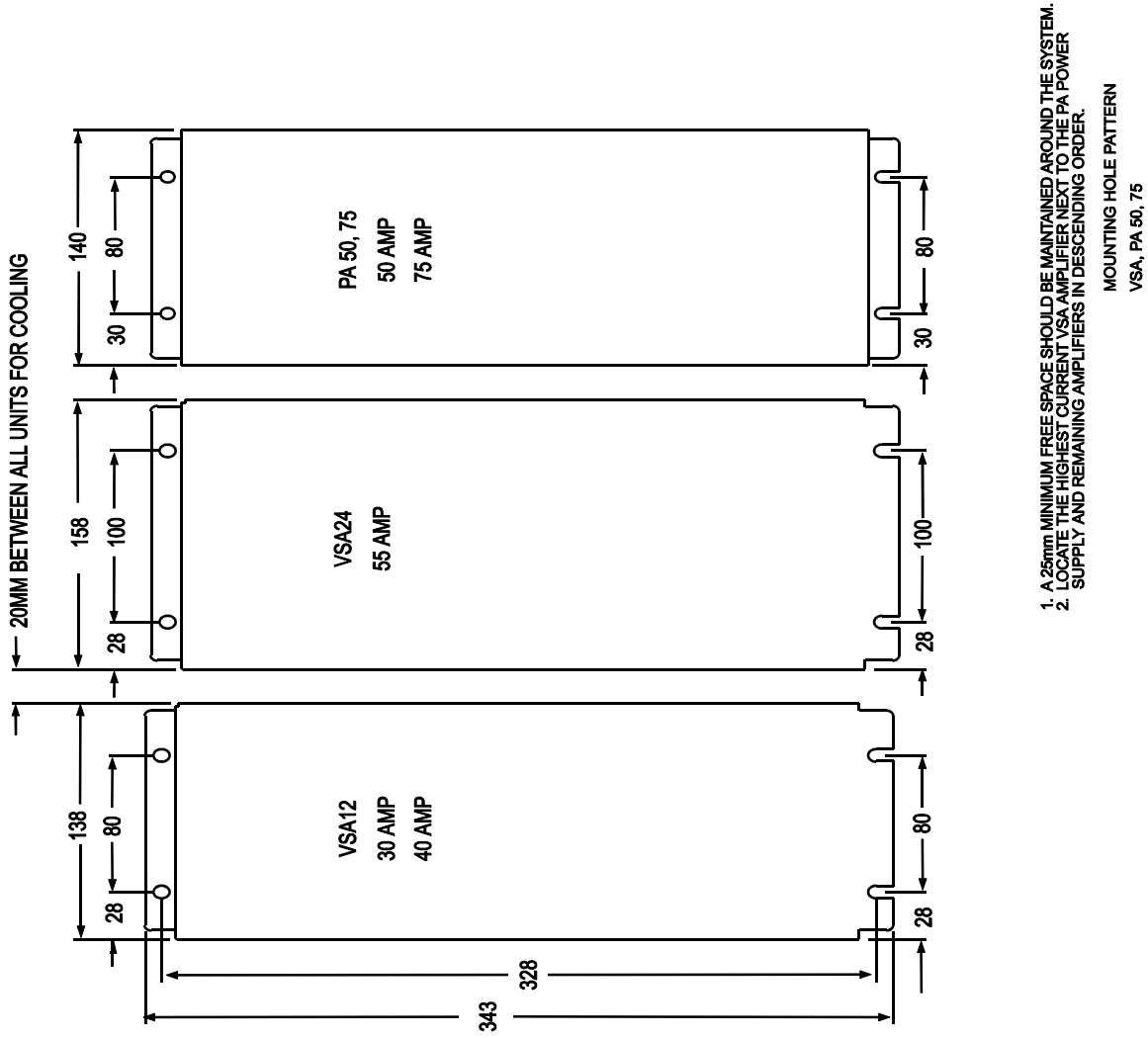


Figure B.8 VSA PA 50/75 Mounting Hole Pattern

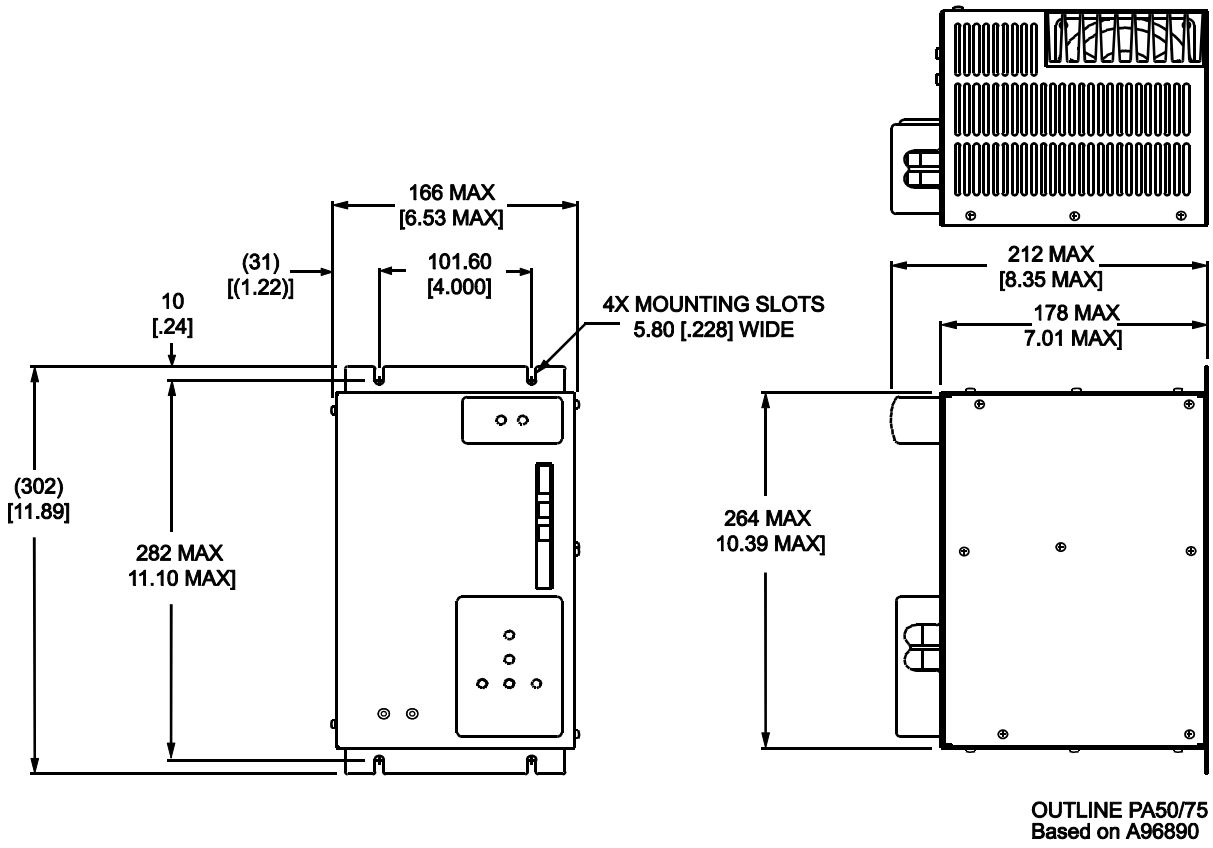


Figure B.9 PA 50/75 AMP, Outline and Dimensions

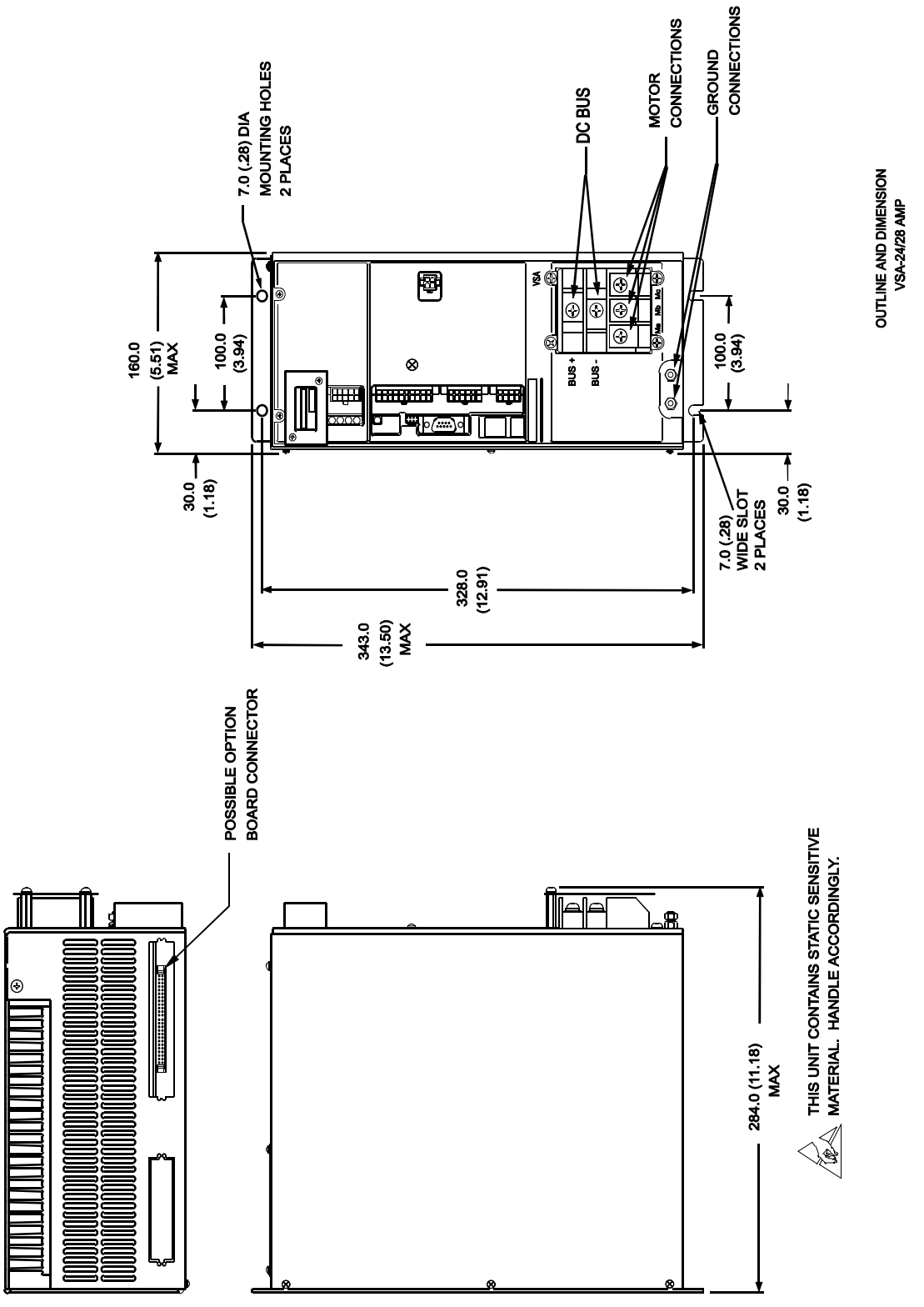


Figure B.10 VSA 24/28 AMP Outline and Dimensions

APPENDIX C
ERROR CODES

C.1 INTRODUCTION

The VECTOR**STAR**'s response to an error depends on the error's severity. There are four levels of severity, listed below in increasing order:

1. Errors that cause warnings.
2. Errors that cause a program break and stop motion, in addition to Level 1 actions.
3. Errors that cause the system to disable and set the FAULT Hardware Output, in addition to Level 2 actions.
4. Errors that disable almost all VECTOR**STAR** functions (including communications) and flash the CPU LED to indicate the error number. These are called firmware errors.

See Chapter 6 for more information about error severity. The following is a complete list of errors generated by the VECTOR**STAR**.

C.2 HARDWARE FAULTS

C.2.1 Firmware Faults

ERROR 2 ***"HARDWARE- U-P FAIL"*** ***SEVERITY 4***

The microprocessor cannot pass self-test. This fault causes the microprocessor to blink the CPU light twice and then pause. The VECTOR**STAR** will not communicate or run the user program. Contact the factory.

ERROR 3 ***"HARDWARE-CHECKSUM"*** ***SEVERITY 4***

The microprocessor cannot pass the checksum self-test. This fault causes the microprocessor to blink the CPU light three times and then pause. The VECTOR**STAR** will not communicate or run the user program. Contact the factory.

ERROR 4 ***"SOFTWARE WATCHDOG"*** ***SEVERITY 4***

The microprocessor has failed the software watchdog self-test. This fault causes the microprocessor to blink the CPU light four times and then pause. The VECTOR**STAR** will not communicate or run the user program. Contact the factory.

ERROR 5 ***"+5 VOLTS"*** ***SEVERITY 4***

The +5 volts is too low. This fault causes the microprocessor to blink the CPU light five times and then pause. The VECTOR**STAR** will not communicate or run the user program. Check the +10 VDC input into the VECTOR**STAR** (Connector C4, Pin 4 or 8). If it is below 6.5 Volts for even a short time, this error will occur. This happens when the logic supply is loaded too heavily, or when the line voltage (PA Connector C1, Pins 2 and 3) is below 98 VAC (115 VAC less 15%).

C.2.2 VECTORSTAR Faults

ERROR 10 ***"REMOTE OFF"*** ***SEVERITY 2***

You attempted to execute an instruction that requires the hardware input REMOTE on the signal connector to be active. This error breaks program execution.

ERROR 11 ***"OVER-TEMP"*** ***SEVERITY 3***

The thermostat on the VECTOR**STAR** heatsink opened, indicating overheating. Overheating may be caused by excessive ambient temperature, obstructed airflow, broken fan, etc. Correct any such condition before resuming operation. REMOVE ALL POWER BEFORE CHECKING THIS. If everything is functioning properly, a drive with a higher current rating may be required. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 12 ***"OVER-CURRENT"*** ***SEVERITY 3***

The VECTOR**STAR** detected an overcurrent. This can be caused by a shorted motor winding, a shorted power transistor, or a short circuit in the wiring. Be sure to check all wiring before resuming operation. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 13 ***"OVER-SPEED"*** ***SEVERITY 3***

The VECTOR**STAR** determined that the speed of the motor was greater than the variable VOSPD. If this occurs occasionally, it may be a nuisance fault that should be corrected by raising VOSPD by 5% or 10%. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 14 ***"POWER BUS"*** ***SEVERITY 3***

The power supply high voltage bus has either an overvoltage fault or an undervoltage fault. See Chapter 3 for more information. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 15 ***"COMP BOARD"*** ***SEVERITY 3***

You attempted to enable the VECTOR**STAR** with the compensation board removed. Replace the compensation board. This error breaks program execution.

ERROR 17 ***"FEEDBACK LOSS"*** ***SEVERITY 3***

The VECTOR**STAR** has detected that one or more wires to the resolver have been broken, or the resolver connector has been removed. This error breaks program execution.

ERROR 18 ***"BAD TL"*** ***SEVERITY 3***

The VECTOR**STAR** has two boards: a small MC board and a larger IBD board. Both boards have the current and voltage rating encoded and they must match. If this error occurs because you exchanged the MC card, then you should replace the original card. If it occurs for some other reason, contact the factory. This error breaks program execution.

ERROR 19 ***"MOTION (HDWR LINE)"*** ***SEVERITY 2***

The MOTION input was off at the beginning of a motion instruction, or it turned off during a motion instruction. This signal comes from the optional I/O card. This error breaks program execution.

ERROR 20 ***"TUNE FAILED"*** ***SEVERITY 3***

The TUNE command failed. Either the inertia on the motor is too large for the desired bandwidth, the motor is not functioning properly, the bus voltage is too low, or the VECTOR**STAR** is not functioning properly. Try reducing the desired bandwidth to correct this problem. Make sure REMOTE is on. If this does not work, attempt to tune the system by hand as described in Chapter 4.

ERROR 21 ***"NOT w/PI"*** ***SEVERITY 3***

Auto tune is not available for PI control (PDF=0).

ERROR 22 ***"+/- 12 VOLTS"*** ***SEVERITY 3***

The ±12 volts is out of tolerance. Contact the factory. This error breaks program execution.

C.2.3 Positioner Faults

ERROR 23 ***"SOFTWARE OVERTRAVEL"*** ***SEVERITY 2***

Software travel limits are enabled and either PMAX or PMIN (the software limits) have been exceeded. If your application does not need software travel limits, or if you want to disable software travel limits temporarily, type:

```
PLIM OFF
```

This error breaks program execution.

ERROR 24 ***"HARDWARE OVERTRAVEL"*** ***SEVERITY 3***

The VECTOR**STAR** detected an overtravel condition while it was enabled. You can print the state of the overtravel limit switch by typing:

```
P LIMIT
```

If LIMIT is 0, then an overtravel condition exists. LIMIT should be connected to a limit switch that has contacts that are normally closed but which open where an overtravel condition occurs. Hardware overtravel limits cannot be disabled. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 25 ***"PE OVERFLOW"*** ***SEVERITY 3***

The variable PE, the position error, exceeded the variable PEMAX. This is also called a following error overflow. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 26 ***"PFB ROLLOVER"*** ***SEVERITY 3***

The variable PFB, the position feedback, exceeded +/-2,147,483,647 counts. If you are using position units, then PFB exceeded the position unit equivalent of +/-2,147,483,647 counts. This can occur if the motor rotates indefinitely in one direction. If your application requires this, consider using the rotary mode as described in Chapter 10.

ERROR 27 ***"R/D JUMPERS"*** ***SEVERITY 3***

Either the jumpers on your VECTOR**STAR** MC2 card are incorrectly set or the wrong TL has been loaded. Contact the factory.

C.3 MOTION ERRORS

C.3.1 Position Calculation Errors

ERROR 30 ***"TOO MANY MOVES"*** ***SEVERITY 2***

You typed in too many move commands (MA, MI, MCGO) from the interactive mode. You can have one move executing and the other pending. The error does not occur when move commands are executed from the user program, because the VECTOR**STAR** sees that the motion buffer is full and delays execution to prevent the error. This error breaks program execution.

ERROR 31 ***"TOO MANY MRD MOVES"*** ***SEVERITY 2***

You attempted to execute a motion instruction that required the profile buffer to be empty. This occurs when two MRD instructions are active at once. You should use a synchronizer to delay the execution of the instruction that caused the error. This error breaks program execution.

ERROR 32 ***"ACC/DEC TOO LOW"*** ***SEVERITY 2***

You entered a motion command that calculated a motion profile where either the acceleration or deceleration segment was more than 30 seconds long. You must increase ACC or DEC or reduce the speed change of the move. This error breaks program execution.

ERROR 33 ***"VEL OUT OF BOUNDS"*** ***SEVERITY 2***

You entered a motion command where the commanded velocity was out of the allowable range. The range for Jog (J) commands is ±VMAX. The range for other motion commands is 0 to +VMAX. This error breaks program execution.

C.3.2 Macro Move/JT/JF Errors

ERROR 40 ***"CHANGED DIRECTION"*** ***SEVERITY 2***

You attempted to change direction with an instruction that does not allow direction to change. These instructions include JT, JF and macro moves. This error breaks program execution.

ERROR 41 ***"MOVE NEEDS MOTION"*** ***SEVERITY 2***

You attempted to execute an instruction that requires the motor to be in motion. These instructions include JT, JF, and MCI/MCA with no velocity parameter specified. This error breaks program execution.

ERROR 42 ***"MOVE w/o TIME"*** ***SEVERITY 2***

You attempted to execute a move that required more time than was available. For example, you attempted a JT or macro segment where the final position could not be reached because of acceleration limits. You may have attempted a JT or JF when you were already well beyond the specified position. This error breaks program execution.

ERROR 43 ***"MACRO NOT READY"*** ***SEVERITY 2***

You attempted to execute a macro move (with the MCGO instruction) in which the last segment of the move did not end at zero speed, or the macro-move memory is empty. The macro-move memory is cleared every time the VECTOR**STAR** is turned on. This error breaks program execution.

ERROR 44 ***"MCD w/MACRO MOVING"*** ***SEVERITY 2***

You attempted to insert a macro-move dwell when the previous macro-move segment ended at a speed other than zero. This error breaks program execution.

ERROR 45 ***"MCA ACTIVE"*** ***SEVERITY 2***

You attempted to insert an MCA segment after an MCI segment. This error breaks program execution.

ERROR 46 "MCI ACTIVE" SEVERITY 2

You attempted to insert an MCI segment after an MCA segment in a macro move. This error breaks program execution.

ERROR 47 "MCI/MCA TOO COMPLEX" SEVERITY 2

You attempted to execute a macro move that required too many segments. This error breaks program execution.

ERROR 48 "MCA/MCI RUNNING" SEVERITY 2
You attempted to build a macro move while another macro move was running. This error breaks program execution.

C.4 SOFTWARE ERRORS

C.4.1 Programming Modes or Motion Modes

ERROR 50 "DRIVE INHIBITED" SEVERITY 2

You attempted to execute an instruction that required the VECTOR**STAR** to be enabled while it was inhibited. This error will break program execution if the instruction is issued from the user program.

ERROR 51 "DRIVE ENABLED" SEVERITY 2

You attempted to execute an instruction that required the VECTOR**STAR** to be inhibited while it was enabled. This error will break program execution if the instruction is issued from the user program.

ERROR 52 "NOT FROM TERMINAL" SEVERITY 1

You attempted to execute an instruction from the terminal that is not allowed from the terminal. This error generates no action.

ERROR 53 "NOT FROM PROGRAM" SEVERITY 1

You attempted to execute an instruction from the program that is not allowed from the program. This error breaks program execution.

ERROR 54 "NOT FROM MONITOR" SEVERITY 1

You attempted to execute an instruction while in the Monitor mode that is not allowed from the Monitor mode. This error generates no action.

ERROR 55 "NOT FROM RECOVERY" SEVERITY 2

You attempted to execute an instruction from the error recovery (the user's error handler or "ERROR$") that is not allowed. This includes attempting to enable the VECTOR**STAR**, GOSUB, and GOTO. This error breaks execution.

ERROR 56 "NOT w/GEAR" SEVERITY 2

You attempted to execute an instruction when the Gear mode was enabled that is not allowed with the Gear mode. For example, MRD, MA, JT, and JF are not allowed with the Gear mode on. This error breaks execution if the instruction was issued from the program.

ERROR 57 "NOT w/PROFILE" SEVERITY 2

You attempted to execute an instruction that is not allowed while the VECTOR**STAR** is profiling. Profiling occurs when move instructions (MA, MI, MRD) or macro moves are executing. Other examples of this are the traverse segment before the accel/decel portion of position dependent jogs (JT, JF), and the accel/decel portions of all jogs (J, JT, JF). This error breaks execution.

ERROR 58 ***"NOT w/JOGGING"*** ***SEVERITY 2***

You attempted to execute an instruction that is not allowed when the VECTOR**STAR** is jogging. This error breaks execution if the instruction was issued from the program.

ERROR 59 ***"NOT w/ROTARY"*** ***SEVERITY 2***

You attempted to execute an instruction that is not allowed when the VECTOR**STAR** is in the Rotary mode. Type:

```
ROTARY OFF
```

to turn the Rotary mode off. This error breaks execution if the instruction was issued from the program.

ERROR 60 ***"OUTSIDE PROTARY"*** ***SEVERITY 2***

You attempted to make an absolute move (either MA or MCA) beyond PROTARY. For example, if PROTARY is 1000 and you typed:

```
MA 2000
```

Use incremental moves (MI and MCI) if you want to move beyond the rotary limit. This error breaks execution if the instruction was issued from the program.

ERROR 61 ***"NORMALIZE FIRST"*** ***SEVERITY 2***

You attempted to turn on the Rotary mode when PFB was less than zero or greater than PROTARY. Use the NORM command to normalize the position to between 0 and PROTARY. This error breaks execution if the instruction was issued from the program.

ERROR 62 ***"RD ALREADY IN USE"*** ***SEVERITY 2***

You attempted to execute RD when RD was in use from some other task. This error occurs when two task levels attempt to simultaneously use the RD command. This error breaks program execution.

ERROR 63 ***"NOT AT THIS LEVEL"*** ***SEVERITY 2***

You attempted to execute a command that is not allowed at the present task level. For example, GOSUB and GOTO are not allowed from within an alarm. This error breaks program execution.

ERROR 64 ***"BACKWARD REGULATION"*** ***SEVERITY 3***

The external input counted backwards more than 30,000 counts when REG was on. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 65 ***"RECORD NOT READY"*** ***SEVERITY 3***

You entered a PLAY command when nothing had been recorded since the last time the VECTOR**STAR** powered up.

C.4.2 Improper Use of Labels

ERROR 70 ***"LABEL NOT FOUND"*** ***SEVERITY 2***

You attempted to branch to a label (either from RUN, GOSUB, or GOTO) that does not exist. This error breaks program execution.

ERROR 71 ***"LABEL USED TWICE"*** ***SEVERITY 2***

The user program has a label that is used more than once. This error breaks program execution.

ERROR 74 "ERROR$ MUST BE LAST" SEVERITY 2

The user's error (ERROR$) must be the last label in the program buffer. You cannot have labels after ERROR$, nor can you use the GOTO or GOSUB commands when the VECTOR**STAR** is executing the error handler. The error handler is intended to provide a graceful exit during error conditions and cannot be used to restart the program. You can use the IF, TIL, and ? commands to execute conditional commands in the error handler. This error breaks program execution.

C.4.3 Invalid Instructions or Entries

ERROR 79 "BAD FORMAT" SEVERITY 2

You entered a format that the VECTOR**STAR** does not recognize. For example, you may have entered:

```
INPUT "INPUT X1" X1[.3]
```

In this case, the decimal point (following the "[") is incorrect. Pay careful attention to the rules for formats. This error breaks program execution if the instruction is issued from the user program.

ERROR 80 "INVALID INSTRUCTION" SEVERITY 2

You attempted to execute an instruction or change a variable that the VECTOR**STAR** does not recognize. This error breaks program execution if the instruction is issued from the user program.

ERROR 81 "NOT PROGRAMMABLE" SEVERITY 2

You attempted to change a variable that is not programmable. This error will break program execution if the instruction is issued from the user program.

ERROR 82 "BAD NUMBER ENTRIES" SEVERITY 2

The instruction that is executing has too many or too few parameters. Look up the instruction in Appendix E to determine the correct number of entries. This error breaks program execution if the instruction is issued from the user program.

ERROR 83 "BAD OR OUT OF RANGE" SEVERITY 2

You entered a parameter to an instruction that was too large or too small. Check Appendix F for limits on variables. This error can also occur when a parameter is in the wrong format, such as a character string where a number is expected. This error breaks program execution if the instruction is issued from the user program.

ERROR 84 "OUT OF BOUNDS" SEVERITY 2

The variable listed is out of bounds. If the variable is protected (that is, set by the factory as defined in Appendix F), contact the factory. If the variable is not protected, set it within its bounds. This error breaks execution.

ERROR 85 "BAD INDIRECTION" SEVERITY 2

You attempted an indirect reference to a user variable that does not exist. For example:

```
X1 10000
P X(X1)
```

X(X1) refers to user variable X10000, which does not exist. The "P X(X1)" will generate this error. This error breaks program execution if the instruction is issued from the user program.

ERROR 86 "USER PROGRAM FULL" SEVERITY 2

You attempted to load a program larger than the VECTOR**STAR** can hold. This occurs with the >BDS instruction and from the Motion Link communications software "Program Transmit (^T)." This error breaks program execution.

ERROR 87 "EMBEDDED QUOTE" SEVERITY 2

You entered a command with an embedded quote. A space must precede an opening quote and follow a closing quote. For example:

```
P"BAD COMMAND"
```

has an embedded quote after the "P." This error breaks program execution if the instruction is issued from the user program.

ERROR 88 "NO CLOSING QUOTE" SEVERITY 2

You entered a command with an odd (as opposed to even) number of quotes. This error breaks program execution if the instruction is issued from the user program.

ERROR 89 "NOT FOR ALARM/HOLD/RECORD" SEVERITY 2

You have specified a switch that is not an allowable switch for an alarm or a hold or record command. For example:

```
A$ REMOTE ON ;ERROR—REMOTE NOT ALLOWED FOR ALARMS
```

This line causes Error 89 since REMOTE is not allowed to fire an alarm.

ERROR 90 "TOO MANY POINTS" SEVERITY 2

You specified too many points in a RECORD command. Only 1000 points total can be recorded. For example, if you are recording four variables, they can be recorded no more than 250 times, since 4*250 = 1000.

C.4.4 Math Errors

ERROR 92 "ZERO DIVIDE" SEVERITY 2

You attempted to divide a number by 0. This error breaks program execution if the instruction is issued from the user program.

ERROR 93 "MATH OVERFLOW" SEVERITY 2

The final result of a calculation or an intermediate result during the calculation of an expression was greater than 231 or less than -231. This error breaks program execution.

ERROR 94 ">2 PARENTHESES" SEVERITY 2

The VECTOR**STAR** evaluated an expression with more levels of parentheses than the VECTOR**STAR** supports. Up to two levels of parentheses are allowed. This error breaks program execution.

ERROR 95 "UNEVEN PARENTHESES" SEVERITY 2

The VECTOR**STAR** encountered an expression in which the number of closing parentheses was not equal to the number of opening parentheses. This error breaks program execution.

ERROR 96 ***"SCALING OVERFLOW"*** ***SEVERITY 2***

During a conversion to or from user units, the result was greater than 231 or less than -231. This error breaks program execution if the instruction is issued from the user program.

ERROR 97 ***"GEAR OVERFLOW"*** ***SEVERITY 3***

The VECTOR**STAR** encountered an overflow when calculating the velocity from the external pulse input. This can be caused when the variable GEARI is too small or GEARO is too large. That is, the input times the ratio of GEARO/GEARI was greater than the highest allowable input frequency, 2 MHz. This error breaks program execution and disables the VECTOR**STAR**.

C.4.5 Communication Errors

ERROR 103 ***"BAUD RATE"*** ***SEVERITY 1***

The variable BAUD contains a value that is not supported by the VECTOR**STAR**. This error occurs during the autobaud sequence and so is never printed to the terminal. You will only see it in the error history buffer. This error has no action.

ERROR 104 ***"ABAUD & MULTIDROP"*** ***SEVERITY 1***

This error is caused by attempting to autobaud while in multidrop communications, which is not allowed. The variable ABAUD is on, indicating request for autobaud, and the variable ADDR is not zero, indicating multidrop communications. This error occurs during the autobaud sequence and so is never printed to the terminal. You will only see it in the error history buffer. This error has no action.

ERROR 105 ***"SERIAL WDOG"*** ***SEVERITY 3***

The serial port did not receive a valid command for WTIME milliseconds when the serial watchdog was enabled (that is, WATCH = 1). This error breaks program execution and disables the VECTOR**STAR**.

C.4.6 Password Errors

ERROR 110 ***"EDIT PASSWORD"*** ***SEVERITY 1***

You attempted to execute an instruction that requires the Editor password. This occurs with the >BDS command. In this case, you must follow the command with the password.

ERROR 111 ***"FACTORY SETTABLE"*** ***SEVERITY 2***

You attempted to change a variable that is protected. These variables are set at the factory. This error breaks program execution if the instruction is issued from the user program.

C.4.7 Errors From IF, TIL and GOSUB Commands

ERROR 115 ***"IF w/o ENDIF"*** ***SEVERITY 2***

The program executed an IF command to begin an IF BLOCK, but could not find the corresponding ENDIF to end the IF block. This error breaks program execution.

ERROR 116 ***"IF NOT STARTED"*** ***SEVERITY 2***

An ELSE, ELIF, or ENDIF was encountered when there was no IF. This will occur, among other times, if you use a GOTO to branch to the middle of an IF/ELIF/ELSE/ENDIF block. This error breaks program execution.

ERROR 117 ***"TIL FOLLOWS ?/TIL"*** ***SEVERITY 2***

The ? or TIL instruction was used to execute a conditional TIL. This error breaks program execution.

ERROR 118 ***"TOO MANY GOSUBS"*** ***SEVERITY 2***

The last GOSUB was one GOSUB too many. The VECTORSTAR has 4 levels of subroutines. This error breaks program execution.

ERROR 119 ***"RETURN w/o GOSUB"*** ***SEVERITY 2***

The VECTOR**STAR** encountered a RET when it was not expecting one. This occurs when there are more returns than GOSUBs. This error breaks program execution.

C.4.8 Power-Up Marker (Not An Error)

ERROR 199 ***"DRIVE POWERED UP"*** ***N/A***

This is not a true error. ERROR 199 is used to mark the error history buffer when the VECTOR**STAR** powers-up.

C.4.9 Internal Errors

ERROR 200 ***"FOLDBACK OUT"*** ***SEVERITY 3***

The factory set variables that control foldback are out of bounds. Contact the factory. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 201 ***"SLIP TOO BIG"*** ***SEVERITY 3***

The induction motor variables that control slip are out of bounds. Contact the factory. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 202 ***"USER PROGRAM CORRUPT"*** ***SEVERITY 3***

The user program is corrupt. Usually, this problem is caused by installing a new battery back-up RAM. This can also occur if power to the VECTOR**STAR** is lost while editing the program. This error will break program execution.

ERROR 203 ***"AMPS BAD"*** ***SEVERITY 3***

The VECTOR**STAR** variable AMPS is invalid. Contact the factory. This error breaks program execution.

ERROR 204 ***"PROGRAM OVERRUN"*** ***SEVERITY 3***

This is an internal error. Contact the factory. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 205 ***"MBUF OVRRUN"*** ***SEVERITY 3***

This is an internal error. Contact the factory. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 206 ***"PROFILE OVERFLOW"*** ***SEVERITY 3***

This is an internal error. Contact the factory. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 208 ***"GENERAL INTERNAL"*** ***SEVERITY 3***

This is an internal error. Carefully write down the entire line that is printed with the error and contact the factory. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 209 ***"STACK OVERFLOW"*** ***SEVERITY 3***

This is an internal error. Carefully write down the entire line that is printed with the error and contact the factory. This error breaks program execution and disables the VECTOR**STAR**.

ERROR 211-219 ***"INTERNAL 1-9"*** ***SEVERITY 3***

These are internal errors. Contact the factory. These errors break program execution and disable the VECTOR**STAR**.

ERROR 255 ***"UNKNOWN"*** ***SEVERITY 3***

This is an internal error. If this error exists in the error history upon initial power-up, clear it with ERR CLR. Contact the factory if this error occurs during operation. This error breaks program execution and disables the VECTOR**STAR**.

APPENDIX D
CUSTOMER SUPPORT

Kollmorgen is committed to quality customer service. Our goal is to provide the customer with information and resources as soon as they are needed. This one number provides order status and delivery information, product information and literature, and application and field techncial assistance.

Note: If you are unaware of your local sales representative, please contact us at the number below. Visit our web site for MotionLink software upgrades, technical articles, and the most recent version of our product manuals.

Kollmorgen Customer Support Network
203 Rock Road Suite A
Radford, VA 24141
Phone: (888) 774-KCSN (5276)
Fax: (540) 639-1640 Inside Sales
Fax: (540) 639-1574 Technical Support
Email: servo@Kollmorgen.com
Http://www.Kollmorgen.com

APPENDIX E
ASCII TABLE

The chart on the following pages is an ASCII Code and Hexadecimal conversion chart. The VECTOR**STAR** does not support extended ASCII (128-255).

ASCII CODE AND HEX CONVERSION CHART

00 NUL 0	10 DLE ^ P 16	20 SP 32	30 0 48	40 @ 64	50 P 80	60 ` 96	70 p 112
01 SOH ^ A 1	11 DC1 ^ Q 17	21 ! 33	31 1 49	41 A 65	51 Q 81	61 a 97	71 q 113
02 STX ^ B 2	12 DC2 ^ R 18	22 “ 34	32 2 50	42 B 66	52 R 82	62 b 98	72 r 114
03 ETX ^ C 3	13 DC3 ^ S 19	23 # 35	33 3 51	43 C 67	53 S 83	63 c 99	73 s 115
04 EOT ^ D 4	14 DC4 ^ T 20	24 $ 36	34 4 52	44 D 68	54 T 84	64 d 100	74 t 116
05 ENQ ^ E 5	15 NAK ^ U 21	25 % 37	35 5 53	45 E 69	55 U 85	65 e 101	75 u 117
06 ACK ^ F 6	16 SYN ^ V 22	26 & 38	36 6 54	46 F 70	56 V 86	66 f 102	76 v 118
07 BEL ^ G 7	17 ETB ^ W 23	27 ‘ 39	37 7 55	47 G 71	57 W 87	67 g 103	77 w 119
08 BS ^ H 8	18 CAN ^ X 24	28 (40	38 8 56	48 H 72	58 X 88	68 h 104	78 x 120
09 HT ^ I 9	19 EM ^ Y 25	29) 41	39 9 57	49 I 73	59 Y 89	69 i 105	79 y 121
0A LF ^ J 10	1A SUB ^ Z 26	2A * 42	3A : 58	4A J 74	5A Z 90	6A j 106	7A z 122
0B VT ^ K 11	1B ESC ^ [27	2B + 43	3B ; 59	4B K 75	5B [91	6B k 107	7B { 123
0C FF ^ L 12	1C FS ^ \ 28	2C , 44	3C < 60	4C L 76	5C \ 92	6C l 108	7C | 124
0D CR ^ M 13	1D GS ^] 29	2D - 45	3D = 61	4D M 77	5D] 93	6D m 109	7D } 125
0E SO ^ N 14	1E RS ^ ^ 30	2E . 46	3E > 62	4E N 78	5E ^ 94	6E n 110	7E ~ 126
0F SI ^ O 15	1F US ^ _ 31	2F / 47	3F ? 63	4F O 79	5F _ 95	6F o 111	7F DEL 127

ASCII CODE AND HEX CONVERSION CHART (CONTD)

80 128	90 144	A0 160	B0 176	C0 192	D0 208	E0 224	F0 240
81 129	91 145	A1 161	B1 177	C1 193	D1 209	E1 225	F1 241
82 130	92 146	A2 162	B2 178	C2 194	D2 210	E2 226	F2 242
83 131	93 147	A3 163	B3 179	C3 195	D3 211	E3 227	F3 243
84 132	94 148	A4 164	B4 180	C4 196	D4 212	E4 228	F4 244
85 133	95 149	A5 165	B5 181	C5 197	D5 213	E5 229	F5 245
86 134	96 150	A6 166	B6 182	C6 198	D6 214	E6 230	F6 246
87 135	97 151	A7 167	B7 183	C7 199	D7 215	E7 231	F7 247
88 136	98 152	A8 168	B8 184	C8 200	D8 216	E8 232	F8 248
89 137	99 153	A9 169	B9 185	C9 201	D9 217	E9 233	F9 249
8A 138	9A 154	AA 170	BA 186	CA 202	DA 218	EA 234	FA 250
8B 139	9B 155	AB 171	BB 187	CB 203	DB 219	EB 235	FB 251
8C 140	9C 156	AC 172	BC 188	CC 204	DC 220	EC 236	FC 252
8D 141	9D 157	AD 173	BD 189	CD 205	DD 221	ED 237	FD 253
8E 142	9E 158	AE 174	BE 190	CE 206	DE 222	EE 238	FE 254
8F 143	9F 159	AF 175	BF 191	CF 207	DF 223	EF 239	FF 255

This side of the table is provided for Decimal to Hex Conversion.
The VECTOR**STAR** does no support extended ASCII (128-255) Decimal to Hex Conversion.

APPENDIX F
VARIABLE QUICK REFERENCE GUIDE

F.1 INTRODUCTION

This appendix lists all the variables on the VECTOR**STAR**. All variables are shown with the required programming conditions. For example, ABAUD has the programming condition "ALWAYS". This means ABAUD can be changed at any time. Other variables require the VECTOR**STAR** to be enabled or disabled. Others, such as feedback variables, are never programmable. "FACTORY" variables can only be changed at the factory. Factory variables program the VECTOR**STAR** for the particular motor it will be controlling. The MOTOR command changes these variables as necessary for the motor.

F.2 STANDARD VARIABLES

Table F.1 Standard Variables[1]

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
A2D	Choose gear input	Always	None	0,1
AMPS	Drive Amps	Factory	I	
ABAUD	Autobaud On	Always	None	0,1
ACC	Acceleration Rate	Always	ACC	0-AMAX
ACTIVE	Monitor Drive	Never	None	
ADDR	Multidrop Address	Always		0,48-57,65-90
ADEN	ACC Units Denominator	Always	None	Long
AMAX	Acc/Dec Maximum	Disabled	ACC	Long>0
ANUM	ACC Units Numerator	Always	None	long
BAUD	Baud Rate	Always	None	300-19200
CAP	Enable Capture	Always	None	0,1
CAPDIR	Polarity of Capture	Always	None	0,1
CLAMP	Enable Clamp Mode	Always	None	0,1
CYCLE	Start Cycle	Never	None	
DEC	Deceleration Rate	Always	ACC	0-AMAX

[1]See Table F.2 for description of long and short.

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
DEP	Shorten Error Msgs	Always	None	0,1
DIR	On if CW is Positive	Always	None	0,1
EXTLOOP	On for Encoder feedback	Disabled	None	0,1
FAULT	On for BDS5 Fault	Always	None	0,1
FOLD	Monitor Foldback Mode	Never	None	
GATE	Monitor GATE Input	Never	None	
GATEMODE	Enable Gate Mode	Always	None	0,1
GEAR	Enable Gear Mode	Always	None	0,1
GEARI	Input Gear Teeth	Always	None	Short
GEARO	Output Gear Teeth	Always	None	Short>0
HOME	Monitor HOME Input	Never	None	
ICMD	Commanded Current	Never	I	
ICONT	Continuous Current	Factory	I	
IDEN	I Units Denominator	Always	None	Long
IFOLD	Monitor Foldback	Never	I	
ILIM	Set Current Limit	Always	I	1-IMAX
IMAX	Maximum Current	Factory	I	
IMON	Monitor Current	Never	I	
I1-16	Monitor 16 Input Lines	Never	None	
IN	Input Word	Never	None	
INUM	I Units Numerator	Always	None	Long
KC	Low Speed Adjust	Always	None	0-255
KF	Feed-Forward Gain	Always	None	Short>0
KP	Pos Loop Gain	Always	None	Short>0
KPROP	Prop. Vel Loop Gain	Always	None	Short>0
KV	Integrating Vel Loop Gain	Always	None	Short>0
KVI	Integrating Vel Loop Gain	Always	None	Short>0
LIMIT	Monitor LIMIT Input	Never	None	
LOAD	Enable reload of app. Program	As Needed	None	0,1
LPF	Enable Low Pass Filter	Always	None	0,1

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
LPFHZ	Low Pass Filter Freq	Always	Hz	0-500
LSTERR	Last error	Never	None	
LSTLBL	Last label executed	Always	None	
MANUAL	Monitor MANUAL Input	Never	None	
MOTION	Monitor MOTION Input	Never	None	
MULTI	Enable Multi-tasking	Always	None	0,1
N	Special Constant=0	Never	None	
O1-8	Set/Monitor Output Lines	Always	None	0,1
OFF	Special Constant=0	Never	None	
OK2EN	OK to enable BDS5	Never	None	
ON	Special Constant=1	Never	None	
OUT	Set/Monitor Output Word	Always	None	0-255
PCAP	Capture Position	Never	POS	
PCMD	Position Command	Never	POS	
PDEN	POS Units Denominator	Always	None	Long
PDF	Choose PDF or PI control	Disabled	None	0,1
PE	Position Error	Never	POS	
PECLAMP	Clamp Position Error	Always	POS	Short>0
PEMAX	Maximum Position Error	Always	POS	Short>0
PEXT	External Position	Always	POS	Long
PFB	Position Feedback	No Motion	POS	Long
PFNL	Final Position	Never	POS	
PL	Enable Position Loop	Always	None	0,1
PLIM	Enable Soft Limits	Always	None	0,1
PMAX	Soft Upper Limit	Always	POS	Long
PMIN	Soft Lower Limit	Always	POS	Long
PROMPT	Enable Prompts	Always	None	0,1
PROTARY	Rotary Distance	Always	POS	Long
PNUM	POS Units Numerator	Always	None	Long
PRD	Position from R/D	Never	Counts	

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
PROP	Enable Prop. Mode	Always	None	0,1
PTRIP1	Position Trip Point #1	Always	POS	Long
PTRIP2	Position Trip Point #2	Always	POS	Long
PXDEN	Extern. Pos Denominator	Always	None	Long
PXNUM	Extern. Pos Numerator	Always	None	Long
RAMP	Ramp control with gear	Always	None	0,1
READY	Enable Drive	Never	None	
REG	Enable Profile Regulation	Always	None	0,1
REGKHZ	Max Regulation Freq.	Always	kHz	1-2000
REMOTE	Monitor REMOTE Input	Never	None	
ROTARY	Enable Rotary Mode	Always	None	0,1
SAT	Monitor Saturation	Never	None	
SCRV	S-curve Type	Always	None	1-5
SEG	Motion Segment	Never	None	
SERIAL	Monitor Serial Port	Never	None	
SS	Enable Single Step	Always	None	0,1
STATMODE	Select STATUS Type	Always	None	0,1
STATUS	Monitor STATUS Output	Never	None	
TMR1	Standard Timer	Always	Msec	Long>0
TMR2	Standard Timer	Always	Msec	Long>0
TMR3	Standard Timer	Always	Msec	Long>0
TMR4	Standard Timer	Always	Msec	Long>0
TRC	Enable Trace	Always	None	0,1
TRIP	Enable Trip Points	Always	None	0,1
TRIP1	Trip #1 Indicator	Never	None	0,1
TRIP2	Trip #2 Indicator	Never	None	0,1
TQ	Enable Torque Loop	Always	None	0,1
VAVG	Averaged VFB	Never	VEL	
VCMD	Velocity Command	Never	VEL	
VDEFAULT	MI/MA Default Velocity	Always	VEL	<VMAX

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
VDEN	VEL Units Denominator	Always	None	Long
VE	Velocity Error	Never	VEL	
VEXT	External Velocity	Never	VEL	
VFB	Velocity Feedback	Never	VEL	
VMAX	Maximum Speed	Factory	VEL	
VNUM	VEL Units Numerator	Always	None	Long
VOFF	Gearbox Velocity Offset	Always	VEL	Long
VOLTS	Drive Voltage	Factory	Volts	
VOSPD	Overspeed Setpoint	Disabled	VEL	Long
VXAVG	Averaged VEXT	Never	VEL	
VXDEN	External Vel Denominator	Always	None	Long
VXNUM	External Vel Numerator	Always	None	Long
WATCH	Enable Serial Watchdog	Always	None	0,1
WTIME	Serial Watchdog Timeout	Always	Msec	Short>0
X1-X250	User Variables	Always	None	Long
XS1-XS50	User Switches	Always	None	0,1
X(X1-X250)	User Indirect Vars	Always	None	Long
Y	Special Constant=1	Never	None	
ZERO	Enable ZEROing Mode	Always	None	0,1

Table F.2 Description of Program Limits

Long Limit	-2147483648	<	x	<	2147483647
Long>0 Limit	0	<	x	<	2147483647
Short Limit	-32768	<	x	<	32767
Short>0 Limit	0	<	x	<	32767

F.3 INTERNAL VARIABLES

The following variables are internal variables and are not normally used by customers. They are set at the factory and program theVECTOR**STAR** for the particular motor it will be controlling. The Motor command changes these variables as necessary for the motor.

Table F.3 Internal Variables

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS
A1-A16	Internal		
ADVSLIP	Internal		
ADVSPD	Internal		
ADVLD	Internal		
ANGLD	Internal	Factory	none
BSLIP	Inductn Base Slip	Factory	mHz
FOLDD	Foldback Delay	Factory	sec/100
FOLDR	Foldback Reset	Factory	sec/100
FOLDT	Foldback Const	Factory	sec/100
IBASE	Inductn Base Amps	Factory	I
IMAG	Induc Mag Current	Factory	I
IND	Select Induction	Factory	None
IZERO	Zeroing Current	Factory	I
MADV	Enable Manual Adv	Factory	None
MANG	Internal		
MSLIP	Manual Slip	Factory	None
POLES	Motor Poles	Factory	Poles*128
SGOOSE	Induction Angle	Factory	None
SLIP	Induction Slip	Never	None
SLOPE	Inductn Slip Slope	Factory	1/10%
VADVTBL	Angle Table Max	Factory	VEL
VBASE	Inductn Base Speed	Factory	VEL

APPENDIX G
SOFTWARE COMMANDS

G.1 EXPRESSIONS AND SYMBOLS

<Label>$ One or two digits followed by a dollar sign. When using GOSUB or GOTO, a user variable can be used as <Label> if its value is between 0 and 99.

<Time> Specifies time in milliseconds. Must be between 0 and 2,147,483,647 (about 25 days).

<Logical> One of the following: GT, GE, LT, LE, EQ, or NE for greater-than, greater-than-or-equal-to, less-than, less-than-or-equal-to, equal-to, or not-equal-to, respectively.

<Expr> Any valid math expression. Valid math expressions include user variables, indirect references to user variables, constants, algebraic and logical math operations, parenthesis.

Examples of valid expressions are:

X1*X2*X3

(X2-VFB)/VOFF
X1&07FH
PFB-PCMD
TMR1/100
(X1+X2)*(X1+(X2-X3)

<Position> Any valid expression for position. The result is assumed to be in position units. The range is +/-2,147,483,647 counts. If your system has position units, then the limits are the position unit equivalent of +/-2,147,483,647.

<Velocity> Any valid expression for velocity. The results is assumed to be in velocity units.

<Traverse> Any valid expression for velocity. The result is assumed to be in velocity units. Traverse is used in macro-moves as the middle speed for three speed moves.

<End> Any valid expression for velocity. The result is assumed to be in velocity units. End is used in macro moves as the end speed of two and three speed moves.

<Text> <Text> is any text string of characters. The control character symbol (^) converts the succeeding character to a control character.

{} Indicates an optional parameter.

Constants - ON, OFF, Y, and N ON and Y are equivalent to 1. OFF and N are equivalent to 0. The constants can be used in any expression and in response to the Input command.

G.2 COMMANDS

The following commands are the instructions used to program the VECTOR**STAR**

; Comment. Comments can follow any instruction. Also, entire lines can be comments. The semicolon must be preceded by a space unless it is the first character in a line. Allowed on any line including the VECTOR**STAR** Editor.

```
GOTO 5 ;THIS IS A COMMENT FOR A COMMAND
;THIS ENTIRE LINE IS A COMMENT
```

$ Labels. Labels can be 0-500 and cannot be repeated. They must be decimal constants. They are allowed only from the user program. The following labels are special purpose labels:

A$	A alarm label
B$	B alarm label
C$	C alarm label
VARIABLE$	variable input label
POWER-UP$	power-up label
AUTO$	AUTO label
MANUAL$	MANUAL label
ERROR$	error handler label
BACKGROUND$	background label

Alarm labels require that you specify the switch that starts the alarm and the state of the switch (ON or OFF) that should trigger the alarm. If the switch is in the specified state when execution is enabled, the alarm will be fired. Otherwise, the alarm is edge sensitive. Specifying ON is actually specifying the positive edge.

Format: <Label>$
<Alarm Label>$ <Switch> <On/Off>

Example:

```
55$
BACKGROUND$
A4 I1 ON
```

?

Quick If. Conditionally executes one instruction if the condition is true, and another instruction if the condition is false. Allowed from the interactive and monitor modes, and the user program.

Format: ?<Condition> {Instruction} {:} {Instruction}

Example:

```
? PFB GT 100 P PFB
? X1 EQ 1 P "X1 = 1" : P "X1 <> 1"
?X1*X2 NE X4|(X5+5) B
? LIMIT EQ ON : P "LIMIT IS OFF"
```

<Condition> is the same as <Expr> <Logical> <Expr>.

<Instruction> is any instruction except TIL.

B

Break program execution. Allowed from the user program or the monitor mode.

Format: B

CONTINUE

Continue motion at the present speed. Turn REG and GEAR off. Optionally, you can specify the number of milliseconds, up to 1 second, that you want the present speed averaged over. If this time is not specified, the speed is averaged over 1 millisecond.

Format: CONTINUE
CONTINUE <time>

Example:

```
CONTINUE 100        ;AVERAGE SPEED FOR .1 SEC.
```

DUMP

Display all the variables and the user program on the terminal, or display the version. Allowed from interactive. Drive must be disabled.

Format:

```
DUMP                ;Dump variables and program
DUMP VERSION        ;Dump firmware version
```

D

Delay program execution for a specified amount of time, up to 2,147,483,647 milliseconds or 25 days. D is an idling command (that is, if you are using multi-tasking, D suspends the task but lets other tasks proceed). Allowed only from the user program.

Format: D <Time>

Example:

D 1000 ;DWELL FOR 1 SECOND

DIS Disable the VECTORSTAR. This command turns off the variable READY. Refer to **Drawing C-84732 for more information. Allowed from the interactive mode, monitor mode, and user program.

Format: DIS

ED Edit the user program. Allowed only from the interactive mode.

Format: ED

Editor Commands:

DEL	Delete a line
F	Find string
C	Change String
I	Enter insert mode
P	Go to a line and print it
NEW	Clear user program
SIZE	Show remaining program memory
PASS	Change password
Empty Line	Go to the next line and print
Escape Key	Exit the insert mode/editor

ELIF Part of block if. Conditionally begins block execution. Allowed from the user program. (See the IF command.)

Format: ELIF <Expr> <Logical> <Expr>

Example:

ELIF PFB GT 100

<Expr> <Logical> <Expr> is the condition.

ELSE Part of block if. Begins last block execution. Allowed from the user program. (See the IF command.)

Format: ELSE

EN Enable the VECTORSTAR. This command turns on the variable READY. Refer to Drawing **C-84732 for more information. Allowed from the interactive mode, monitor mode, and user program.

Format: EN

END

End a task. If you are using multi-tasking, END ends that task. If there are no special labels present in the program (except POWER-UP$), then END is equivalent to Break (B). If there are special labels, the VECTORSTAR becomes inactive waiting for a task to resume execution.

Format: END

ENDIF

Part of block if. Ends block if. Allowed from the user program. (See the IF command).

Format: ENDIF

ERR

Display an error message, display the error history, or clear the error history. Allowed from the interactive and monitor modes and user program.

Format: ERR <Error Number>
ERR <Option>

Where <Error Number> is a valid error number and <Option> can be HIST or CLR.

Example:

```
ERR 25          ;DISPLAY MESSAGE FOR ERR 25
ERR HIST        ;DISPLAY ERROR HISTORY
ERR CLR         ;CLEAR ERROR HISTORY
```

GOSUB

Go to a subroutine. Allowed only from the user program.

Format: GOSUB <Label>

Example:

```
GOSUB 25
GOSUB X3
```

GOTO

Go to a program label. Allowed only from the user program.

Format: GOTO <Label>

Example:

```
GOTO 25
GOTO X5
```

H

Delay (Hold-up) execution of a task until a switch is in the specified state. You can use any switch except REMOTE and XS11-XS50 (XS1-XS10 are allowed). H is an idling command; if you are using multi-tasking, H suspends the task but lets other tasks proceed. Allowed only from the user program.

Format: H <Switch> <ON/OFF>

Example:

```
H XS1 ON
H I1 OFF
```

IF

Conditionally execute a block of instructions. Allowed from the user program.

Format: IF <Expr> <Logical> <Expr>

Example:

```
IF PFB GT 100
...                    ;FOLLOW WITH ELSE, ETC.

IF X1*X2 NE X4|(X5+5)
...                    ;FOLLOW WITH ELSE, ETC.
```

INPUT

Prompt the operator for an input variable. If limits are specified, then make sure operator stays within them. If they are not specified, then use the limits of the variable being prompted for. W is an idling command (that is, if you are using multi-tasking, INPUT suspends the task until the operator presses the enter key, but lets other tasks proceed). Allowed only from the user program.

Format: INPUT "<Text>" <Variable>{decimal}{Min}{Max}

Where <Variable> is any valid, programmable variable. You can optionally specify maximum and minimum limits (if you include one, you must include the other). {Min} is the minimum input allowed and {Max} is the maximum input allowed.

If you specify decimal, the input received from the operator will be multiplied by 10^{decimal}. The VECTOR**STAR** does not use floating point math internally. The input command allows you to receive floating point input from the operator.

Example:

```
INPUT "ENTER NEW SPEED" X1[3]  -5000 5000
INPUT "ENTER NEW CURRENT LIMIT" ILIM
```

In the first example, if the operator entered 1.234, the VECTOR**STAR** would store 1234.0 in X1; that is, 1.234 is multiplied by 10^3 = 1000. Note that if you specify {decimal}, {Max} and {Min} limit the value after the multiplication. In the above example, {Max} = -5000 limits the operator to -5.000.

J

Jog at a continuous speed. Allowed from the interactive mode and the user program.

Format: J <Velocity>

Example:

J 1000
J X1

JF

Jog, but wait until the Position command (PCMD) crosses the specified position before beginning accel/decel. Speed must not be zero when executing this instruction. Allowed from the interactive mode and the user program.

Format: JF <Position> <Velocity>

Example:

JF 10000 10
JF 100*X1 4000

JT

Jog at a continuous speed, but delay beginning accel/decel so that the Position command will equal the specified position when the accel/decel is complete. Allowed from the interactive mode and the user program.

Format: JT <Position> <Velocity>

Example:

JT -610000 100
JT 100*X45 -800

K

Disable the drive and break the program. Allowed from interactive and monitor modes and the user program. See Drawing **C-84732 for more information.

Format: K

MA

Move to the specified position at the specified speed. If the speed is not specified, it is assumed to be VDEFAULT. Allowed from the interactive mode and the user program.

Format: MA <Position> {Velocity}

Example:

```
MA 10000 1000        ;MOVE AT 1000
MA 0                 ;MOVE TO 0 AT VDEFAULT
```

MCA

Define an absolute macro-move section to the specified position at the specified traverse and ending speeds. See Chapter 8, "General Programming," for descriptions of defaults. Allowed from the interactive mode and the user program.

Format: MCA<Position>{Traverse}{End}

Example:

```
MCA 1000 100 500
MCA 2000 10
MCA 5000
MCA 7000 0
```

MCD

Define a macro-move dwell section for the specified time. This is only valid when the previous macro-move section ended at zero speed. When used with the profile regulation mode, time is inversely proportional to external input frequency. Allowed from the interactive mode and the user program.

Format: MCD <Time>

Example:

```
MCD 500        ;DWELL 0.5 SECONDS
```

MCGO

Execute a macro move. This is only valid when the last macro-move section ended at zero speed. Allowed from the interactive mode and the user program.

Format: MCGO

MCI

Define an incremental macro-move section for the specified distance at the specified traverse and ending speed. See Chapter 8 for descriptions of defaults. Allowed from the interactive mode and the user program.

Format: MCI<Position>{Traverse}{End}

Example:

```
MCI 100000 500 5000
MCI 3000 10
MCI -56000
MCI 8000 0          ;LAST SECTION
```

MI

Incrementally move the specified distance at the specified speed. If the speed is not specified, it is assumed to be VDEFAULT. Allowed from the interactive mode and the user program.

Format: MI <Position> {Velocity}

Example:

```
MI 10000 1000       ;MOVE AT 1000
MI -1000            ;MOVE BACK 1000
```

MOTOR

Display the present motor drive combination. This command is used to determine the motor for which your VECTOR**STAR** was configured when it was shipped. This command is not normally used by the customer.

Format: MOTOR

MRD

Make an absolute move so that the output of the Resolver-to-Digital converter output (PRD) will equal the specified value. A direction option indicates whether the motion should be clockwise (CW), counterclockwise (CCW), or whichever way is shortest (no option specified). Allowed from the interactive mode and the user program.

Format: MRD <R/D-Position> <Velocity> {Option}

Where R/D-Position is greater than 0 and less than the resolution of the Resolver-to-Digital (R/D) converter. For the standard 12-bit resolution of R/D converter, the upper limit is 4095. Option is either CCW or CW.

Example:

```
MRD 3200 100 CCW    ;MOVE CCW AT 100 RPM
MRD 0 50            ;GO BEST WAY AT 50 RPM
```

NORM

Normalize the Position command and position feedback to the specified position. Allowed from the interactive mode and the user program when there is no commanded motion.

Format: NORM <Position>

Example:

```
NORM 1000
```

P

Print the variables specified with optional formats on a new line. Allowed from the interactive and monitor modes and the user program.

Format: P <Expr> {format} | "<Text>" ...

Where {format} is the print format specifying field width and Hex output. The ellipsis (...) indicates that the P can be followed by up to 15 different expressions and text strings.

Format can be:		
	B	Binary
	H	Hex
	S	ON or OFF
	C	ASCII Character
	Blank	Decimal Integer
	nn.m	Floating Point Output where nn is the total number of digits; m is the number of digits after the decimal point.
	nn.m.p.	Same as nn.m except only print p digits after the decimal point (p must be less than m).

Examples:

```
P PFB VFB IMON          ;PRINT 3 FEEDBACK VARS
P PFB[4]                ;PRINT PFB IN 4 CHARS
P IN[H]                 ;PRINT INPUT IN HEX
P IN[5H]                ;PRINT INPUT, 5 HEX CHARS
P 123456[.4]            ;PRINT 12.3456
P 123456[.4.2]          ;PRINT 12.34
P "VECTORSTAR"          ;PRINT VECTORSTAR ON THE SCREEN
P "XPOS=" PFB           ;PRINT PFB WITH TEXT
```

PS

Print with status. This is identical to the P command, except status of the VECTOR**STAR** is displayed on the end of the printed line. See P for format and examples. Allowed from the interactive and monitor modes and the user program.

PLAY

Playback recorded points. This command prints all the variables that were recorded by the last RECORD command. Normally, you should use Motion Links' PLAYBACK,FROM VECTOR**STAR** command rather than the VECTOR**STAR** PLAY command. Motion Link formats, plots, and prints data in a much more readable form than does the VECTOR**STAR**.

R

Refresh screen. This command is the same as the P command except that no line feed is printed. This command can be used to overprint, the practice of refreshing the display by printing a line with new values over the same line with old values. It is generally used for status updating. See P for examples and formats. Allowed from the interactive and monitor modes and the user program.

RD

Delay program execution for a specified period of time, but use external clock to time the delay. REG need not be on for RD to function properly. Allowed only from the user program.

Format: RD <Time>

Example:

```
RD 1000
```

RECORD

Record 1-4 variables for a specified period of time. This command allows you to record most VECTOR**STAR** variables in real time for later playback. You cannot record PE, REMOTE, TMR1, TMR2, TMR3, TMR4, VAVG, VXAVG, or any user switches. Allowed from the user program or from the interactive mode.

Format: RECORD <Number> <Time> <1 to 4 Variables>

Where <Number> is the number of intervals over which the variables will be recorded,

and <Time> is the time in milliseconds of each interval.

Note: <Number> <= 1000 for 1 Variable
<Number> <= 500 for 2 Variables
<Number> <= 333 for 3 Variables
<Number> <= 250 for 4 Variables

Examples:

```
RECORD 1000 1 VFB                ;RECORD VFB ONCE/MSEC FOR
                                 ;1 SECOND
RECORD 500 10 VCMD VFB           ;RECORD VCMD AND VFB ONCE/10
                                 ;MSEC FOR 5.0 SECOND
RECORD 100 1000 VCMD VFB PCMD    ;RECORD VCMD, VFB, AND PCMD
                                 ;ONCE/SECOND FOR 100 SECONDS
```

RET

Return from a subroutine. Allowed only from the user program.

Format: RET

RS

Refresh screen with status. This command is identical to the R command, except status of the drive is displayed at the end of the printed line. See P for format and examples. Allowed from the interactive and monitor modes and the user program.

RUN

Run a program starting at the specified label. Allowed from the interactive mode. If no label is specified, run multi-tasking.

Format: RUN <Label>
RUN ;RUN MULTI-TASKING

Example:

```
RUN 4
RUN X1
RUN
```

S

Stop motion using a deceleration of AMAX. Allowed from the interactive and monitor modes and the user program.

Format: S

TIL

Continuously execute an optional instruction until condition is true. If no instruction is specified, then delay program execution until the condition is true. <Instruction> cannot be another TIL. Allowed only from the user program.

Format: TIL <Expr> <Logical> <Expr> [Instruction]

Example:

```
TIL PFB GT 100 P PFB
TIL X1*X2 NE X4/(X5+5) GOSUB 100
TIL VFB LT 100 ;DELAY EXECUTION
```

TUNE

Tune the motor to a new load. This command is used if the motor needs to be re-tuned. The tuning parameters (KP, KV, KVI, and KPROP) determine the motor stability and response time. Often when the motor load is changed, tuning parameters need to be reset. The Tune command specifies Bandwidth and Stability. Higher bandwidth will produce faster response time. Higher stability will produce less overshoot, but noisier performance. Allowed from the interactive mode and the user program. Only available when PDF = 1.

Format: TUNE <Bandwidth> <Stability>

Where Bandwidth is 5, 10, 15,...50 Hz and stability is 1, 2, or 3.

Example:

```
TUNE 25 2
```

W

Wait for a specified motion profile segment to start before continuing program execution. W is an idling command (that is, if you are using multi-tasking, W suspends the etask but lets other tasks proceed). Allowed only from the user program.

Format: W

Examples:

```
W 3          ;WAIT FOR SEGMENT 3 TO START
W 0          ;WAIT FOR MOTION TO STOP
```

ZPE Clear the position error. This command is useful when enabling the position loop when position error has been allowed to accumulate. Allowed from the interactive and monitor modes and the user program.

<BDS Send (download) a program from the VECTOR**STAR** Program Memory to the terminal. Allowed from the interactive mode and the user program.

Format: <BDS

>BDS Receive (upload) a program from the terminal and store it in VECTOR**STAR** program memory. A password may be specified. If the editor password has been set and the password is incorrect or not specified, then an error will result and the original program memory will remain. Allowed form the interactive mode and the user program.
Format: >BDS {PASS}

where PASS is the password as set in the editor.

Example:

```
>BDS SECRET      ;UPLOAD, PASSWORD=SECRET
>BDS             ;UPLOAD, NO PASSWORD
```

APPENDIX H
COMMAND TIMINGS

This appendix gives approximate timings of representative commands. Command times are difficult to predict because they depend on many factors, including whether the VECTOR**STAR** is enabled, whether profile motion has been commanded, whether electronic gearbox or profile regulation have been enabled and so on. The times listed here are based on these conditions:

1. The VECTOR**STAR** is enabled.

2. PLIM, PL, and LPF are on.

3. TQ, and PROP are off.

4. No profiles are being calculated. That is, the VECTOR**STAR** is enabled, but not in motion.

5. GEAR and REG are off.

Acceleration profiles increase the execution time by 40%-50%. If the GEAR mode is enabled, increase execution time by 10% to 20%. Profile regulation increases execution time by as much as 20%. As you can see, if either gear or profile regulation is enabled, and the VECTOR**STAR** is executing the acceleration or deceleration portion of a motion profile, then the times can be 60% greater than those shown here. These commands are not meant to represent the worst case but are only provided as an estimate of the execution times.[1]

[1]These times are based on tests run at Industrial Drives Electronic Lab. Reference Test 67 of May 21, 1990.

? 1 EQ 1 O1 ON	1.93 msec
20$	0.72 msec
GOSUB 10 & 10$	0.84 msec
GOTO 10 & 10$	0.84 msec
GOSUB 120 & 120$	1.46 msec

IF 1 EQ 0
X1=1
ELIF 1 EQ 0
X1=2
ELSE
X1=3
ENDIF
	4.88 msec

J 1000	1.37 msec (calc. time only)
JF 40960 1000	2.46 msec (calc. time only)
JT 40960 1000	2.46 msec (calc. time only)
MA 40960	1.96 msec (calc. time only) (VDEFAULT 1000)
MA 40960 1000	2.52 msec (calc. time only)
MCGO MCI 1000 1000 200 MCI 1000 0	5.50 msec (calc. time only)
MI 40960	1.96 msec (calc. time only) (VDEFAULT 1000)
MI 40960 1000	2.52 msec (calc. time only)
MI 40960 1000 (GATEMODE ON)	2.84 msec (calc. time only)
MI 40960 1000 (GEAR ON)	2.98 msec
MI 40960 (GEAR ON)	2.61 msec (VDEFAULT=1000)
MI 40960 1000 (GEAR ON & In Motion)	3.16 msec
MI 40960 1000 (REG ON)	2.84 msec
MI 40960 (REG ON)	2.61 msec (VDEFAULT=1000)
MI 40960 1000 (REG ON & In Motion)	3.14 msec
MRD 1000 100 CW	1.71 msec
NORM 0	1.29 msec
O1 ON	0.95 msec
O1 OFF	0.96 msec
OUT=OUT!0C8H	1.41 msec
O1 ON (GEAR ON)	0.99 msec
P X1	2.09 msec
P X1[8]	2.09 msec
P “X1=” X1	2.53 msec
RET	1.46 msec
TIL 1 EQ 0	1.27 msec
X1=X2	0.85 msec
X1=X2+1	1.07 msec
X1=X2-1	1.07 msec
X1=X2*100	1.16 msec
X1=X2/100	1.18 msec
X1=X2+1 (GEAR ON)	1.12 msec
ZPE	1.27 msec

NOTE: Drive is enabled (EN) for all timing tests.
ACC=DEC=100,000 and SCRV=2

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

Q

R

S

T

U

V

W

X

Y

Z