

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

BDS5

USER'S MANUAL

Old Number M93102 - ISSUE 3

New number MB5001H

TECHNICAL MANUAL CONFIGURATION

(USER'S MANUAL M93102)

PAGE NO. **DESCRIPTION** **# CHANGE 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 - iv ..... Table of Contents 0
v ..... List of Figures 0
vi ..... List of Tables 0
1-1 -- 1-16 ..... Chapter 1 System Description 0
2-1 -- 2-14 ..... Chapter 2 Getting Started 0
3-1 -- 3-42 ..... Chapter 3 Programming Language 0
4-1 -- 4-44 ..... Chapter 4 User Programs 0
5-1 -- 5-10 ..... Chapter 5 Debugging 0
6-1 -- 6-8 ..... Chapter 6 Compensation 0
A-1 -- A-2 ..... Appendix A Warranty Information 0
B-1 -- B-4 ..... Appendix B ASCII Table 0
C-1 -- C-16 ..... Appendix C Software Commands 0
D-1 -- D-14 ..... Appendix D Error Codes 0
E-1 -- E-6 ..... Appendix E Variable Quick Reference 0
F-1 -- F-2 ..... Appendix F Command Timings 0
Glossary-i -- xiv ..... Glossary 0
Index-i -- viii ..... Index 0
-- ..... BDS5 Upgrade Notices 0

* Zero in this column indicates an original page

CONFIGURATION TABLE

(USER'S MANUAL M93102)

RECORD OF REVISIONS

ISSUE NO. (Revision)	DATE	CHANGED PAGES/BRIEF DESCRIPTION OF CHANGE	CHANGE NO.
3	15 Mar 95	Replaces issue dated 15 Feb 95	Original Release

THANK YOU!

KOLLMORGEN
Motion Technologies Group

Thank you and congratulations for choosing Industrial Drives' servo products for your motion control requirements. We seek to provide our customers with quality products, excellent support and outstanding value. In an effort to provide you with dependable and useful documentation, we offer you an opportunity to critique this manual with your comments and suggestions. Your feedback on this reader comments form is very important to us. Please answer the questions below and return the form to:

INDUSTRIAL DRIVES - Technical Manual Department
201 Rock Road
Radford, VA 24141
U.S.A.
FAX: 703/731/0847

Name:_____ Title:_____

Company:_____

Street Address:_____

City:_____ State:_____ Zip:_____

Telephone:_____ Fax:_____

Product:_____

Manual Part Number:_____

Please check the rating that best represents your opinion on each topic.

		Excellent	Good	Fair	Poor
1.	Overall clarity and readability.	❑	❑	❑	❑
2.	Organization of the manual.	❑	❑	❑	❑
3.	Information completeness.	❑	❑	❑	❑
4.	Information accuracy.	❑	❑	❑	❑
5.	Installation procedures.	❑	❑	❑	❑
6.	Ability to quickly find information you need.	❑	❑	❑	❑
7.	Graphics.	❑	❑	❑	❑
8.	Figures (usefulness).	❑	❑	❑	❑
9.	Tables (usefulness).	❑	❑	❑	❑
10.	Overall rating of this manual.	❑	❑	❑	❑

Please list any errors._____

What did you like least about this manual?_____

What did you like most about this manual?_____

How would you improve this manual?_____

Signature:_____ Date:_____

© Copyright 1993, Danaher Motion Kollmorgen. All rights reserved
Printed in the United States of America

NOTICE

Not for use or disclosure outside of Danaher Motion Kollmorgen except under written agreement.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means (electronic, mechanical, photocopying, recording, or otherwise) without the written permission from the publisher. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damager resulting from the use of the information contained herein.

This document is proprietary information of Danaher Motion Kollmorgen, furnished for customer use ONLY. No other uses are authorized without written permission from Danaher Motion Kollmorgen.

Information in this document is subject to change without notice and does not represent a commitment on the part of Danaher Motion Kollmorgen. Therefore, information contained in this manual may be updated without notice due to product improvements, etc., and may not conform in every respect to former issues.

This product is covered by U. S. Patents:

4,447,771
4,479,078
4,490,661
Other (foreign patents pending)

U. L. is a trademark of Underwriter's Laboratories.
N. E. C. is a trademark of the National Electric Code.
Kollmorgen **GOLD**Line, BDS4, BDS5, and PSR4/5 are trademarks of Danaher Motion Kollmorgen.



Dangerous voltages, currents, temperatures, and energy levels exist in this product and in the associated servomotor(s). Extreme caution should be exercised in the application of this equipment. Only qualified individuals should attempt to install, setup, 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

Danaher Motion Kollmorgen

Phone: 1-800-777-3786 or (815) 226-3100
Technical Support Fax: (540) 731-5679

FOREWORD

The commitment to quality at Industrial Drives 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 can now be seen at the Smithsonian which has on display 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 Industrial Drives. 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, Industrial Drives (known in the early days as part of Inland Motor Division of Kollmorgen) has continued to enhance its sophisticated engineering solutions to pioneer new product development.

The results of these and other efforts has 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. Industrial Drives pioneered rare earth magnet development for the servo motor industry.

Between 1974 and 1980, Industrial 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 decade of 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, Industrial Drives continues to lead the way with digital servo positioning capability and our newest motor offering, the GOLDLINE Series, incorporating the very latest high-energy, rare earth magnets (neodymium iron boron). 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, Industrial Drives and its parent, Kollmorgen Corporation, continue to rank first in servo technology.

Other achievements? Yes, too many in fact to mention. Each achievement stands as a testimony to the committed 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 User's Manual is designed to help you properly operate a BDS5 Servo System. You do not have to be an expert in motion control to utilize 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.

The BDS5 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 operate the BDS5 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 BDS5 properly.

These instructions are intended to aid you to administer the BDS5 to your own applications. Your safety and satisfaction are important to Industrial Drives. 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 BDS5.
- **SINGLE BLOCKED** text defines words that are displayed by the BDS5 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

NOTICE:

This manual is the second of a two part manual structure. The Installation and Setup Manual is intended to instruct the user on the installation procedures and practices to be used with the BDS5.

TABLE OF CONTENTS

CHAPTER 1. SYSTEM DESCRIPTION

1.1 Introduction 1-1
1.2 Product Description 1-1
1.3 Features 1-1
1.4 Part Number Description 1-3
1.4.1 BDS5 Model Number 1-4
1.4.2 Compensation Module Model Number 1-5
1.4.3 PSR4/5 Model Number 1-6
1.4.4 ER-External Resistor Kit Model Number 1-7
1.4.5 Molex Assembly Tools 1-7
1.5 Specifications and Ratings 1-8
1.6 Theory of Operation 1-12
1.7 Simplified Schematic Diagram and System Diagram 1-13

CHAPTER 2. GETTING STARTED

2.1 Introduction 2-1
2.2 Computer Requirements 2-1
2.3 Software Installation 2-1
2.3.1 Backing Up the Disk(s) 2-1
2.3.2 Software Installation 2-2
2.3.2.1 Install on a Hard Disk 2-2
2.3.2.2 Install on a Floppy Disk 2-2
2.3.3 Establishing Communications 2-3
2.4 Motion Link Overview 2-4
2.4.1 Menus and Windows 2-4
2.4.1.1 Program 2-4
2.4.1.2 Variables 2-4
2.4.1.3 Capture 2-5
2.4.1.4 Scope 2-5
2.4.1.5 Options 2-5
2.4.1.6 Help 2-6
2.4.1.7 Utilities 2-6
2.4.2 Editor 2-7
2.4.2.1 File 2-7
2.4.2.2 Edit 2-7
2.4.2.3 GOTO 2-8
2.4.2.4 Insert/Delete 2-8
2.4.2.5 Cursor 2-8
2.4.2.6 Help 2-8
2.4.3 Types Of Data Files 2-9
2.4.4 Using IBM-PC Compatibles 2-9
2.5 Motion Link Setup Program 2-9
2.6 Processor Modes 2-9
2.6.1 Prompts 2-9
2.6.2 Descriptions of Modes 2-10
2.6.2.1 Interactive Mode 2-10
2.6.2.2 Run Mode 2-12
2.6.2.3 Monitor Mode 2-12
2.6.2.4 Single-Step Mode 2-12
2.6.2.5 Trace Mode 2-12
2.6.2.6 Other Modes 2-13

CHAPTER 3. PROGRAMMING LANGUAGE

3.1 Introduction 3-1
3.2 Instructions 3-1
3.2.1 Comments 3-1
3.3 Variables 3-1
3.3.1 Variable Units 3-2
3.3.2 Three Types of Variables 3-2
3.3.3 Variable Limits 3-2
3.3.4 Switches 3-2
3.3.5 Printing Variables 3-2
3.3.6 Changing a Variable 3-3
3.3.7 Programming Conditions 3-3
3.3.8 Power-up and Control Variables 3-3
3.3.9 Initial Settings of Control and User Variables 3-4
3.3.10 User Variables 3-7
3.3.10.1 Indirect User Variables 3-7
3.3.11 User Switches 3-8
3.3.12 Special Constants 3-8
3.4 Math 3-8
3.4.1 Hexadecimal 3-8
3.4.2 Algebraic Functions 3-10
3.4.3 Logical Functions: AND, OR 3-9
3.5 General Purpose Input/ Output 3-10
3.5.1 Whole Word I/O 3-10
3.6 Fault Logic 3-11
3.6.1 Firmware Faults, Area 1 3-13
3.6.2 Fault Logic, Area 2 3-13
3.6.3 Fault Latch, Area 3 3-13
3.6.4 Ready Latch, Area 4 3-13
3.6.5 ACTIVE, Area 5 3-13
3.6.6 Relay and STATUS Control, Area 6 3-13
3.6.7 Motor Brake 3-14
3.6.8 Output Relay 3-14
3.7 Drive Control 3-14
3.7.1 Direction Control, DIR 3-14
3.7.2 Position 3-14
3.7.2.1 Position Command and Feedback, PCMD & PFB 3-14
3.7.2.2 Position Error, PE & PEMAX 3-14
3.7.2.3 R/D Position, PRD 3-15
3.7.2.4 Sampling PFB, PCMD and PEXT 3-15

3.7.3 Velocity 3-15
3.7.3.1 VCMD, VFB, VE, & VAVG 3-15
3.7.3.2 Velocity Limits, VMAX & VOSPD 3-16
3.7.4 Current 3-16
3.7.4.1 Motor Current, ICMD & IMON 3-16
3.7.4.2 Current Limits, IMAX & ILIM 3-16
3.7.5 Enabling the Position Loop with PL 3-16
3.7.6 Controlling the Velocity Loop with PROP 3-16
3.7.7 Enabling the BDS5 3-16
3.7.8 Limiting Motor Current 3-17
3.7.8.1 Continuous Current, ICONT 3-17
3.7.8.2 Foldback Current, IFOLD 3-17
3.7.8.3 Monitoring Current Limits 3-18
3.8 Motion Commands 3-18
3.8.1 Basic Motion Commands 3-18
3.8.1.1 AMAX, ACC, & DEC 3-18
3.8.1.2 EN, STOP, & LIMITS 3-18
3.8.1.3 Enabling Motion with MOTION 3-19
3.8.1.4 STOP (S) Command 3-19
3.8.1.5 STOP and BREAK with Control X (^X) 3-19
3.8.2 Limiting Motion 3-19
3.8.2.1 Hardware Travel Limits 3-19
3.8.2.2 Software Travel Limits, PMAX & PMIN 3-20
3.8.2.3 User Position Trip Points, PTRIP1 & PTRIP2 3-20
3.8.3 Profiles 3-20
3.8.3.1 S-Curves 3-20
3.8.3.2 Move Absolute (MA) Command 3-21
3.8.3.3 Move Incremental (MI) Command 3-22
3.8.3.4 Incremental Move Example 3-22
3.8.3.5 Profile Limits 3-22
3.8.3.6 Multiple Profile Commands 3-23
3.8.3.7 Profile Final Position, PFNL 3-23
3.8.4 JOG (J) Command 3-23
3.8.5 NORMALIZE (NORM) Command 3-23
3.8.6 Zero Position Error (ZPE) Command 3-24
3.8.7 MACRO MOVES 3-24
3.8.7.1 MCA, MCI, MCD, & MCGO 3-24
3.8.7.2 Macro Move Example #1 3-25
3.8.7.3 Macro Move Example #2 3-25
3.8.8 R/D BASED MOVE (MRD) Command 3-26
3.8.9 Capturing Position 3-26
3.8.9.1 Enabling Capture, CAP & PCAP 3-26
3.8.9.2 Capture Direction, CAPDIR 3-26
3.8.9.3 Speeding Up Homing Sequences 3-26
3.8.10 Clamping 3-27
3.8.10.1 Clamping and Homing 3-27
3.8.11 JOG TO (JT) & JOG FROM (JF) 3-28
3.8.11.1 Registration 3-29
3.8.11.2 Registration Example 3-29
3.8.11.3 Multiple JF/JT Commands 3-30
3.9.11.4 Changing Profiles During Motion 3-30
3.8.12 External Inputs 3-31
3.8.12.1 Analog Input 3-32
3.8.13 Electronic Gearbox 3-32
3.8.13.1 Gear Ratio, GEARI & GEARO 3-32
3.8.13.2 Gearbox Example 1 3-32
3.8.13.3 Gearbox Example 2 3-33
3.8.13.4 Profiles and Gearbox 3-33
3.8.13.5 Velocity Offset, VOFF 3-35
3.8.13.6 Gearbox, ACC/DEC, and Jogs 3-35
3.8.14 Profile Regulation 3-35
3.8.14.1 REG & REGKHZ 3-35
3.8.14.2 Profile Regulation and Counting Backwards 3-36
3.8.14.3 Regulation Example 3-36
3.8.15 Encoder Feedback 3-37
3.8.16 CONTINUE 3-37
3.9 CONTROL LOOPS 3-37
3.9.1 Position Loop 3-38
3.9.2 Velocity Loop 3-38
3.9.2.1 Proportional Velocity Loop 3-38
3.9.2.2 Integrating Velocity Loop 3-38
3.9.3 Torque Command 3-39
3.9.4 Power-Up Control Loops 3-39

CHAPTER 4. USER PROGRAMS

4.1 Introduction 4-1
4.2 Programming Techniques 4-1
4.2.1 Example Application 4-3
4.2.2 Application Specification 4-3
4.2.3 Application Flowchart 4-3
4.2.4 Commented Program 4-5
4.2.5 Customer Service 4-6
4.3 Editing 4-6
4.3.1 Motion Link Editor 4-6
4.3.2 BDS5 Resident Editor 4-7
4.3.2.1 Editor Print (P) 4-7
4.3.2.2 Next Line 4-7
4.3.2.3 Password (PASS) 4-7
4.3.2.4 INSERT (I) 4-8
4.3.2.5 FIND (F) 4-8
4.3.2.6 CHANGE (C) 4-8
4.3.2.7 DELETE (DEL) 4-9
4.3.2.8 Size 4-9
4.3.2.9 NEW 4-9
4.4 Building A Program 4-10
4.4.1 Basic Commands 4-10
4.4.1.1 Labels 4-10
4.4.1.2 RUN 4-10
4.4.1.3 BREAK (B) 4-10
4.4.1.4 GOTO 4-10
4.4.1.5 GOSUB and RET 4-11
4.4.2 Conditional Commands 4-11
4.4.2.1 Quick IF (?) Command 4-11
4.4.2.2 Nesting ? Commands 4-12

4.4.2.3 TIL Command 4-12
4.4.2.4 IF, ELIF, ELSE, and ENDIF Commands 4-13
4.4.2.5 IF vs. ? 4-14
4.4.2.6 Nesting IF commands 4-15
4.4.2.7 IF's with GOTO and GOSUB 4-15
4.5 Using the General Purpose Inputs 4-15
4.6 Interfacing with the Operator 4-16
4.6.1 PRINT (P) 4-17
4.6.1.1 Printing Decimal Numbers 4-17
4.6.1.2 Printing Decimal Points 4-17
4.6.1.3 Printing Hex Numbers 4-17
4.6.1.4 Printing Binary Numbers 4-18
4.6.1.5 Printing Switches 4-18
4.6.1.6 Printing Expressions 4-18
4.6.1.7 Printing ASCII Characters 4-18
4.6.1.8 Printing Control Characters 4-19
4.6.1.9 Cursor Addressing 4-19
4.6.1.10 Printing BDS5 Status (PS) 4-20
4.6.2 REFRESH (R & RS) Commands 4-20
4.6.3 INPUT 4-20
4.6.3.1 INPUT Limits 4-20
4.6.3.2 INPUT and Decimal Point 4-21
4.6.4 SERIAL Switch 4-21
4.7 Idling Commands 4-21
4.7.1 HOLD (H) 4-21
4.7.2 DWELL (D) 4-22
4.7.3 WAIT (W) 4-22
4.8 MULTI-TASKING 4-23
4.8.1 Multitasking and Autobauding 4-23
4.8.2 MULTI 4-23
4.8.3 END Command 4-23
4.8.4 Enabling and Disabling Multi-tasking 4-23
4.8.5 Idling 4-25
4.8.5.1 Pre-Execution Idle 4-25
4.8.5.2 Post-Execution Idle 4-26
4.8.5.3 Avoiding Idling 4-26
4.8.6 Alarms (Task Levels 1-3) 4-26
4.8.6.1 Restrictions of Alarms 4-27
4.8.6.2 Printing with Alarms 4-27
4.8.7 Variable Input (Task Level 4) 4-27
4.8.7.1 Using Variable Input with Profiles 4-28
4.8.7.2 Restrictions of Variable Input 4-28
4.8.8 Main Program Level (Task Level 5) 4-29
4.8.8.1 Power-Up Routine (POWER-UP$) 4-29
4.8.8.2 Error Handler (ERROR$) 4-29
4.8.8.3 Auto Routine (AUTO$) 4-29
4.8.8.4 Manual Program (MANUAL $) 4-30
4.8.8.5 Typical AUTO/MANUAL Programs 4-30
4.8.9 Background (Task Level 6) 4-30
4.8.9.1 Restrictions of Background 4-32
4.9 Units 4-32
4.9.1 User Units 4-32
4.9.1.1 Current Units 4-32
4.9.1.2 Other User Units 4-33
4.9.1.3 External Units 4-33
4.9.2 Machine Specific Units 4-35
4.9.3 Position Rotary Mode, ROTARY, & PROTARY 4-37
4.9.3.1 Choosing PROTARY, PNUM, and PDEN 4-37
4.9.3.2 Rotary Mode and Absolute Moves 4-38
4.10 Serial Communications 4-38
4.10.1 Autobauding 4-38
4.10.1.1 Setting the BDS5 to Autobaud 4-38
4.10.1.2 Autobauding and MOTION 4-38
4.10.1.3 Enabling Autobaud with ABAUD 4-38
4.10.1.4 Baud Rate, BAUD 4-39
4.10.2 Prompts 4-39
4.10.3 Serial Watchdog 4-39
4.10.4 Transmit/Receive Programs 4-39
4.10.4.1 <BDS Command Receiving from the BDS5 4-39
4.10.4.2 The >BDS Command Transmitting to the BDS5 4-39
4.10.5 System Dump 4-40
4.10.5.1 Version Dump 4-40
4.10.6 Multidrop Communications 4-40
4.10.6.1 Broadcast 4-41
4.11 Program Examples 4-41

CHAPTER 5. DEBUGGING

5.1 Introduction 5-1
5.2 Debugging Modes 5-1
5.2.1 Single-Step 5-1
5.2.2 Trace 5-2
5.2.2.1 Motion Link and Trace 5-2
5.3 Debugging and Multi-Tasking 5-2
5.4 Removing Code 5-3
5.5 Synchronizing Your Program 5-4
5.5.1 Using the Timers, TMR1-4 5-4
5.5.2 Regulation Timer, RD 5-4
5.5.3 Motion Segments 5-4
5.5.4 WAIT (W) 5-5
5.5.5 Gating Motion with GATE 5-6
5.6 Hints 5-6
5.7 Error Log 5-8
5.7.1 Error Levels 5-8
5.7.2 DEP 5-8
5.7.3 Error History 5-9
5.7.4 Displaying Error Messages 5-9
5.7.5 Firmware Errors 5-9

CHAPTER 6. COMPENSATION

6.1 Introduction 6-1
6.2 System Compensation 6-1
6.2.1 Critical Damping 6-2
6.2.2 Underdamping 6-2

6.2.3 Overdamping....6-2
6.2.4 Ringing....6-2
6.3 Tuning....6-3
6.3.1 If Your System Is Completely Unstable....6-3
6.3.2 Reducing ILIM....6-3
6.4 TUNE Command....6-4
6.5 Tuning the Bds5 Yourself....6-4
6.5.1 Tuning the Velocity Loop....6-4
6.5.2 Tuning the Position Loop....6-5
6.6 RECORD and PLAY....6-6
6.7 Problems....6-6
6.7.1 Overloading the Motor....6-6
6.7.2 Compliance....6-7
6.7.3 Non-Linear Mechanics....6-7
6.7.4 Resonance....6-7
6.7.5 Low-Pass Filters....6-8

APPENDIX A. WARRANTY INFORMATION

APPENDIX B. ASCII TABLE

APPENDIX C. SOFTWARE COMMANDS

APPENDIX D. ERROR CODES

APPENDIX E. VARIABLE QUICK REFERENCE

APPENDIX F. COMMAND TIMINGS

LIST OF FIGURES

FIGURE	PAGE
1.1 BDS5 Model Number Scheme	1-4
1.2 Compensation Model Number Scheme	1-5
1.3 PSR4/5 Model Number Scheme	1-6
1.4 External Regen Resistor Model Number Scheme	1-7
2.1 BDS5 Instruction Screen	2-3
2.2 BDS5 State Table	2-11
3.1 BDS5 Enable/Fault Logic Diagram	3-12
3.2 A Simple Profile	3-20
3.3 S-Curve Profile	3-21
3.4 Macro Move Example #1	3-25
3.5 Macro Move Example #2	3-25
3.6 Jog From (JF) Command	3-28
3.7 Jog To (JT) Command	3-29
3.8 BDS5 Master/Slaving	3-34
3.9 BDS5 Control Modes	3-40
4.1 Sample Flowchart	4-4
4.2 Auto/Manual Flowchart	4-31
4.3 Master/Slave Block Diagram	4-34
6.1 Critical Damping	6-2
6.2 Underdamping	6-2
6.3 Overdamping	6-2
6.4 Ringing	6-2

LIST OF TABLES

TABLE	PAGE
1.1 BDS5 Model Number Scheme	1-4
1.2 PSR4/5 Model Number Scheme	1-6
1.3 External Regen Resistor Model Number Scheme	1-7
1.4 Specifications	1-8
1.5 Environmental Specifications	1-12
1.6 Mechanical Specifications	1-12
2.1 Cursor Control Keys	2-8
2.2 BDS5 Rules for Prompts	2-10
2.3 BDS5 Prompts	2-10
2.4 Monitor Mode Commands	2-12
3.1 Standard Units	3-2
3.2 Power-Up State of Programmable Units	3-4
3.3 Rules For Math Expressions	3-9
3.4 Output 1-8 Decimal Values	3-10
3.5 Input 1-16 Decimal Values	3-11
3.6 PRD: Ranges and R/D Resolutions	3-15
3.7 S-Curve Acceleration Chart	3-21
3.8 R/D Converter Accuracy	3-26
3.9 Encoder Resolution	3-37
4.1 BDS5 Conditions	4-11
4.2 Block-IF Restrictions and Options	4-14
4.3 Desired Operation of Program Example	4-14
4.4 Printing BDS5 Status	4-20
4.5 Multi-Tasking Overview	4-24
4.6 How to Enable Multi-Tasking	4-25
4.7 How to Disable Multitasking	4-25
4.8 Four Idling Commands	4-26
4.9 To Execute AUTO$	4-30
4.10 To Execute MANUAL$	4-30
4.11 Common User Units	4-32
4.12 System Resolutions	4-33
4.13 Setting External Units in Master/Slave Systems	4-33
4.14 English Conversion (12-bit R/D Only)	4-35
4.15 Metric Conversion (12-bit R/D Only)	4-35
4.16 External Units Conversion	4-36
4.17 BDS5 Prompts	4-41
5.1 Multi-Tasking Debug Prompts	5-3
5.2 Segments for Different Moves	5-5
5.3 Error Severity Levels and Actions	5-8
6.1 Tuning Criterion	6-1
6.2 Allowed Tune Command Stability Settings	6-4
6.3 Velocity Loop Bandwidth vs. KVI	6-5
6.4 Velocity Loop Bandwidth vs. KP_{MAX}	6-5

CHAPTER 1

SYSTEM DESCRIPTION

1.1 INTRODUCTION

The information in this chapter will enable you to understand the BDS5's basic functions and features. These concepts will allow you to apply them to your own unique applications.

1.2 PRODUCT DESCRIPTION

The BDS5 is a full-featured, high-performance, brushless positioning servo in one compact enclosure —it is the smallest, totally-integrated package available to motion control users. The BDS5 combines a positioner, a servo amplifier, and an I/O interface into one unit. The BDS5 sets new standards for motion control with its simple BASIC-like command structure and sophisticated decision-making capability. The BDS5 provides the outstanding servo performance that you have come to expect from Industrial Drives. Using a high-performance microprocessor, the BDS5 does not have to compromise on either positioner software or servo performance. This single microprocessor closes all servo loops, resulting in a truly integrated positioning system. The BDS5 has the features and performance you need in your next positioning application.

1.3 FEATURES

The BDS5 offers a wide feature set to accommodate real world positioning requirements:

- LOW COST

The BDS5 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 TO INSTALL

The BDS5 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

The BDS5 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. The BDS5 has separate acceleration and deceleration rates, as well as linear, half S-curve, and full S-curve acceleration profiles. The BDS5 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

The BDS5 can precalculate moves to begin motion within one millisecond after a transition on the GATE input. This provides rapid and repeatable motion initiation. The BDS5 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:

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

The BDS5 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 the BDS5 in convenient units, such as feet, inches, miles, RPM, and degrees.

- SUPERIOR SERVO LOOP CONTROL

The BDS5 offers smooth, high-resolution control. Standard BDS5 position repeatability is better than one arc-minute, bidirectional. The BDS5 has a 32-bit position word. The BDS5 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 8000 RPM.

- SELF-TUNING

The BDS5 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 the BDS5 do the rest.

- POWERFUL MICROPROCESSOR

The heart of the BDS5 is the 16-bit processor that delivers high performance. The result: the BDS5 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 the BDS5 features not available in standard velocity drives, such as self-tuning, very low velocity offset, and digitally-adjustable servo tuning parameters. The optional analog input permits you to use the BDS5 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

The BDS5 offers a complete set of error diagnostics. When an error occurs, the BDS5 displays an English

language error message. The BDS5 remembers the last 20 errors even through power loss. In addition, the BDS5 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. The BDS5 offers trace and single-step modes so that you can debug your program. The BDS5 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

The BDS5 has up to 32 I/O sections that you connect via ribbon cable to standard OPTO-22 compatible I/O boards or to INDUSTRIAL DRIVES I/O-32. The I/O-32 provides either fixed 24-volt or removable, industry standard, optically-isolated I/O in a GOLDLINE style package.

- SERIAL COMMUNICATIONS

The BDS5's serial communications provide a powerful link to other popular factory automation devices such as PLC's, process control computers, and smart terminals. The BDS5 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. The BDS5 can autobaud from 300 baud to 19.2k baud, eliminating the need to set dip switches to start communicating.

- MOTION LINK

Industrial Drives also offers MOTION LINK, a powerful, menu-driven communications package for your IBM-PC (c) compatible computer. With this package, the BDS5'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

The BDS5's programming language allows you to write operator-friendly, menu-driven software. By incorporating an INDUSTRIAL DRIVES Data Entry Panel, or any other serial communications device, the operator can be prompted for specific process data.

- MONITOR MODE

The BDS5 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 DESCRIPTION

A model number is printed on a gold and black tag on the front of your BDS5, PSR4/5, 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:

1.4.1 BDS5 Model Number

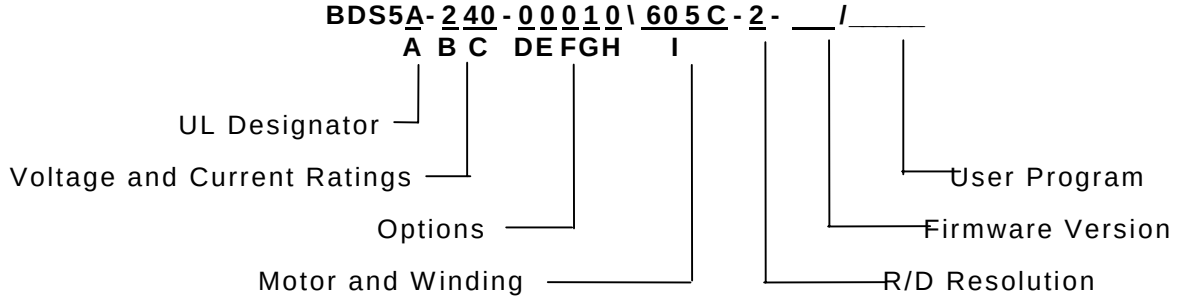


Figure 1.1. BDS5 Model Number Scheme

Table 1.1. BDS5 Model Number Scheme

LEGEND	DEFINITION
A	UL Designator
A	UL Listed (standard)
V	Non - UL Listed
B	Voltage Rating
1	115 VAC
2	230 VAC
C	Current Rating
03	3 Amps/Phase
06	6 Amps/Phase
10	10 Amps/Phase
20	20 Amps/Phase
30	30 Amps/Phase
40	40 Amps/Phase
55	55 Amps/Phase
D	Mechanical Options (0 indicates standard feature)
0	Standard
E	Communication Options
0	RS-232 (standard)
1	RS-422/RS-485
F	Input Options
0	Encoder Input (standard)
1	Analog Input
2	Pulse Input
9	No Input
G	I/O Options
0	8 I/O (standard)
1	32 I/O
H	R/D Accuracy Options
0	8 ARC min (standard)
I	Motor and Winding
Motor and Winding	Specifies Motor Model Type, Winding
R/D Resolution	R/D Resolution
2	12-Bit (4096 counts/rev)
4	14-Bit (16384 counts/rev)
Firmware Version	Firmware Version (Assigned by Industrial Drives, not normally specified when ordering) Most current firmware supplied -- unless otherwise specified.
User Program	User Program (This is reserved for systems that are programmed by Industrial Drives. This is not normally specified when ordering.)

1.4.2 Compensation Module Model Number

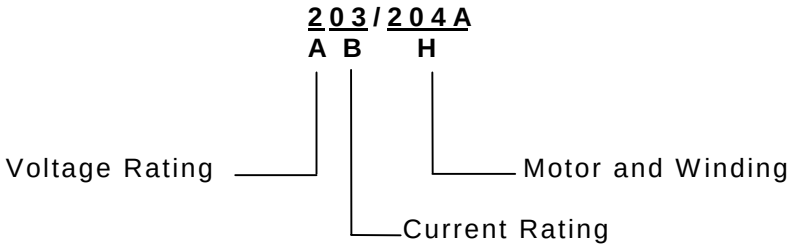


Figure 1.2. Compensation Model Number Scheme

A partial model number is printed on a gold and black tag on the front of the compensation module (the black plastic box secured to the front of your BDS5 by two screws). 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 BDS5. It is important that the motor, the BDS5, and the compensation module model numbers all agree. For example, if your BDS5 model number is

BDS5-203-00000\204A-1-101/2021,

then your compensation module model number must be:

203/204A,

and your motor must be a 204A. An example of a 204A motor model number is:

B-204-A-21.

CAUTION

YOU MUST HAVE THE PROPER COMPENSATION MODULE INSTALLED FOR YOUR MOTOR AND BDS5. 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 BDS5, the motor, or both.

1.4.3 PSR4/5 Model Number

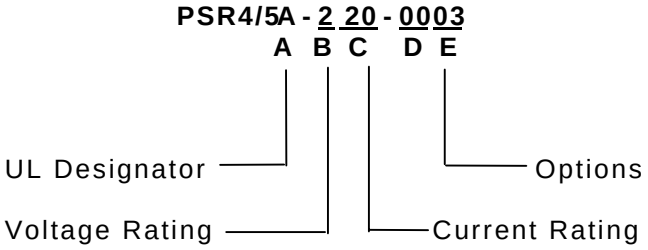


Figure 1.3. PSR4/5 Model Number Scheme

Table 1.2. PSR4/5 Model Number Scheme

LEGEND	DEFINITIONS
A	UL Designator
A	UL Listed (standard)
V	Non - UL Listed
B	Voltage Rating
1	115 VAC
2	230 VAC
C	Current Rating
12	12 Amps/Phase
20	20 Amps/Phase
50	50 Amps/Phase
75	75 Amps/Phase
D	Mechanical Options
00	No Option (standard)
E	Electrical Regen Options for 12 and 20 Amp Models Only
00	Standard Internal 40 W Regen (standard)
01	External Regen (230 VAC Only) 8.8 Ohms, 400 W., Requires ER-01 Resistor Kit
02	External Regen (115 VAC Only) 5.5 Ohms, 200 W., Requires ER-02 Resistor Kit
03	External Regen (230 VAC Only) 5.8 Ohms, 700 W., Requires ER-03 Resistor Kit
E	Electrical Regen Options for 50 and 75 Amp Models Only
00	No internal shunt regeneration (standard) Requires external regeneration resistor kit ER-2X

1.4.4 ER-External Resistor Kit Model Number

NOTE

Contact Industrial Drives Application Engineering to size regeneration capability.

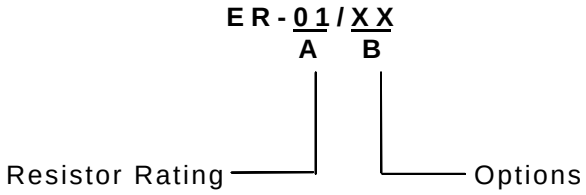


Figure 1.4. External Regen Resistor Model Number Scheme

Table 1.3. External Regen Resistor Model Number Scheme

LEGEND	DEFINITIONS
A	Resistor Rating
01	8.8 Ohms, 400 W., 230V, 12 & 20 Amp Models Only
02	5.5 Ohms, 200 W., 115V, 12 & 20 Amp Models Only
03	5.8 Ohms, 700 W., 230V, 12 & 20 Amp Models Only
20	4.5 Ohms, 500 W., 230V, 50 & 75 Amp Models Only
21	4.4 Ohms, 1000 W., 230V, 50 & 75 Amp Models Only
22	2.2 Ohms, 1000 W., 230V, 75 Amp Models Only
23	2.2 Ohms, 2000 W., 230V, 75 Amp Models Only
B	Options
00	None available at this printing.

1.4.5 Molex Assembly Tools

GOLDLINE series electronics (BDS4's, BDS5's, and PSR4/5's) use Molex MINI-FIT JR. series connectors. The necessary connectors and pins are included in your BDS5 and PSR4/5 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

1.5 SPECIFICATIONS AND RATINGS

Table 1.4. Specifications

BDS5-1XX-(90-160 VAC L-L OUTPUT TO MOTOR)				
DESCRIPTION	**BDS5-103**	**BDS5-106**	**BDS5-110**	**BDS5-120**
Main DC Bus Minimum Maximum	 130 VDC 225 VDC	 130 VDC 225 VDC	 130 VDC 225 VDC	 130 VDC 225 VDC
Unregulated Logic Bus	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS
Output Current (RMS/∅) Convection Cooled (45°C AMB) Continuous (RMS) Peak (2.0 sec) (RMS)	 3.0 AMPS 6.0 AMPS	 6.0 AMPS 12.0 AMPS	 10.0 AMPS 20.0 AMPS	 Fan Cooled 20.0 AMPS 40.0 AMPS
Output KVA (@ 160 VDC Bus) Continuous (45°C AMB) Peak (2.0 sec) (RMS)	 0.6 KVA 1.2 KVA	 1.2 KVA 2.4 KVA	 2.0 KVA 4.0 KVA	 4.0 KVA 8.0 KVA
Internal Heat Dissipation	30 WATTS	40 WATTS	60 WATTS	110 WATTS
PWM Switching Frequency	10.0 kHz	10.0 kHz	10.0 kHz	10.0 kHz
Motor Current Ripple Frequency ±10%	 20.0 kHz	 20.0 kHz	 20.0 kHz	 20.0 kHz
Resolver Excitation Frequency	8.5 kHz	8.5 kHz	8.5 kHz	8.5 kHz
Form Factor RMS/AVG	≤ 1.01	≤ 1.01	≤ 1.01	≤ 1.01
Fan (115 VAC)	N/A	N/A	N/A	0.2 AMPS

Table 1.4. Specifications (Cont.)

BDS5-2XX-(160-253 VAC L-L OUTPUT TO MOTOR)				
DESCRIPTION	**BDS5-203**	**BDS5-206**	**BDS5-210**	**BDS5-220**
Main DC Bus Minimum Maximum	 225 VDC 360 VDC	 225 VDC 360 VDC	 225 VDC 360 VDC	 130 VDC 225 VDC
Unregulated Logic Bus	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS	±15-20 VDC @0.25 AMPS +8-12 VDC @1.00 AMPS
Output Current (RMS/∅) Convection Cooled (45°C AMB) Continuous (RMS) Peak (2.0 sec) (RMS)	 3.0 AMPS 6.0 AMPS	 6.0 AMPS 12.0 AMPS	 10.0 AMPS 20.0 AMPS	 20.0 AMPS 40.0 AMPS
Output KVA (@ 160 VDC Bus) Continuous (45°C AMB) Peak (2.0 sec) (RMS)	 1.2 KVA 2.4 KVA	 2.0 KVA 4.0 KVA	 4.0 KVA 8.0 KVA	 8.0 KVA 16.0 KVA
Internal Heat Dissipation	35 WATTS	50 WATTS	75 WATTS	150 WATTS
PWM Switching Frequency	10.0 kHz	10.0 kHz	10.0 kHz	10.0 kHz
Motor Current Ripple Frequency ±10%	 20.0 kHz	 20.0 kHz	 20.0 kHz	 20.0 kHz
Resolver Excitation Frequency	8.5 kHz	8.5 kHz	8.5 kHz	8.5 kHz
Form Factor RMS/AVG	≤ 1.01	≤ 1.01	≤ 1.01	≤ 1.01
Fan (115 VAC)	N/A	N/A	N/A	0.2 AMPS

Table 1.4. Specifications (Cont.)

PSR4/5-1XX-(90 - 160 VAC L-L INPUT)		
DESCRIPTION	**PSR4/5-112-**	**PSR4/5-120-**
Main AC Line Input Voltage	90-160 VAC	90-160 VAC
Phase	1-3	1-3
Frequency	47-63 Hz	47-63 Hz
Current Cont. (RMS) 3-Phase Single-Phase	12.0 AMPS 10.0 AMPS	20.0 AMPS 16.0 AMPS
Peak (2.0 sec) 3-Phase Single-Phase	24.0 AMPS 20.0 AMPS	40.0 AMPS 32.0 AMPS
Peak (50.0 msec) 3-Phase Single-Phase	50.0 AMPS 42.0 AMPS	80.0 AMPS 64.0 AMPS
Control AC Line Input Voltage	90-132 VAC	90-132 VAC
Phase	1	1
Frequency	47-63 Hz	47-63 Hz
Main DC Bus Output Voltage (Nominal 115 VAC Input) Current 115 VAC	 160 VDC 1.1 AMPS RMS	 160 VDC 1.1 AMPS RMS
Regeneration Shunt Resistor (Internal)	15 OHM	7.5 OHM
Shunt Regulator Current (PK)	15.3 AMPS	30.6 AMPS
Power Dissipation (Cont.)	40 WATTS	40 WATTS
Power Dissipation (PK)	3.5 KW	7.0 KW
Internal Heat Dissipation	120 WATTS	120 WATTS
Regeneration Shunt Resistor (External Min)	5.5 OHM	5.5 OHM
Shunt Regulator Current (PK)	41.8 AMPS	41.8 AMPS
Power Dissipation (Cont.)	200 WATTS	200 WATTS
Power Dissipation (PK.)	9.6 KW	9.6 KW
Soft-Start Surge Current (Max)	80 AMPS	80 AMPS
Charge Time (Max)	25 MSEC	25 MSEC

Table 1.4. Specifications (Cont.)

PSR4/5-2XX-(160 - 253 VAC L-L INPUT)		
DESCRIPTION	**PSR4/5-212-**	**PSR4/5-220-**
Main AC Line Input Voltage	160 - 253 VAC	160 - 253 VAC
Phase	1-3	1-3
Frequency	47-63 Hz	47-63 Hz
Current Cont. (RMS) 3 Phase Single Phase	12.0 AMPS 10.0 AMPS	20.0 AMPS 16.0 AMPS
Peak (2.0 sec) 3 Phase Single Phase	24.0 AMPS 20.0 AMPS	40.0 AMPS 32.0 AMPS
Peak (50.0 msec) 3 Phase Single Phase	50.0 AMPS 42.0 AMPS	80.0 AMPS 64.0 AMPS
Control AC Line Input Voltage	90-132 VAC	90-132 VAC
Phase	1	1
Frequency	47-63 Hz	47-63 Hz
Main DC Bus Output Voltage (Nominal 115 VAC Input) Current 115 VAC	 325 VDC 1.1 AMPS RMS	 325 VDC 1.1 AMPS RMS
Regeneration Shunt Resistor (Internal)	25 OHM	12 OHM
Shunt Regulator Current (PK)	15 AMPS	30 AMPS
Power Dissipation (Cont.)	40 WATTS	40 WATTS
Power Dissipation (PK)	5.6 KW	11.2 KW
Internal Heat Dissipation	120 WATTS	150 WATTS
Regeneration Shunt Resistor (External Min)	8.8 OHM	8.8 OHM
Shunt Regulator Current (PK)	44.3 AMPS	44.3 AMPS
Power Dissipation (Cont.)	400 WATTS	400 WATTS
Power Dissipation (PK)	17.3 KW	17.3 KW
Soft Start Surge Current (Max)	150 AMPS	150 AMPS
Charge Time (Max)	25 MSEC	25 MSEC

Table 1.5. Environmental Specifications

Operating Temperature*: 3, 6, & 10 AMP Units (Convection Cooled) 20 Amp Units (Internal Fan Cooled)	 0^o C to 45^o C 0^o C to 45^o C
Storage Temperature	-20^o C to 70^o C
Humidity (Non-Condensing)	10% to 90%

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

Table 1.6. Mechanical Specifications

MODEL NUMBER	WIDTH		HEIGHT		DEPTH	
	MM	IN.	MM	IN.	MM	IN.
BDS5-X03-	56	2.20	340	13.49	280	11
BDS5-X06-	76	2.99	340	13.49	280	11
BDS5-X10-	98	3.86	340	13.49	280	11
BDS5-X20-	98	3.90	340	13.49	280	11
PSR4/5-X12 & X20-	76	3.00	340	13.49	280	11

1.6 THEORY OF OPERATION

Drawing D-93030 shows a system overview.

- MICROPROCESSOR SYSTEM

The BDS5 is a digital positioner and servo drive combined into one unit. The velocity loop is 100% digital. The BDS5 has battery backup RAM to remember your program and most variables through power-down.

- RESOLVER-TO-DIGITAL CONVERTER

The BDS5 is based on a Resolver-to-Digital (R/D) converter. The R/D generates a tachometer signal for your convenience. However, the BDS5 does not use the analog tach signal.

- SERIAL PORT

The BDS5 has a serial port for communications. This port allows you to monitor the operation, issue commands, and transmit a program.

- DISCRETE INPUTS

The BDS5 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 BDS5 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 BDS5 has 10 discrete outputs. Notice that O1 appears both on Connector C2 with optical isolation and on Connector C8.

- ENCODER INPUT

The BDS5 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 BDS5.

- ENCODER EQUIVALENT OUTPUT

The BDS5 provides encoder format output derived from the R/D converter.

- ANALOG INPUT (OPT1 CARD)

As an option, the BDS5 can accept a ±10 volt analog input. This input is converted to digital format by the BDS5. Gain and offset adjustments are made digitally inside the BDS5, not with potentiometers.

- PULSE INPUT (OPT2 CARD)

The BDS5 can accept special pulse inputs. The standard BDS5 can accept signals directly from encoders or encoder-like devices. As an option, the BDS5 can accept other pulse formats, such as count/direction or up/down.

- LED'S

The BDS5 provides LED's for diagnostics. These LED's are on the front panel of the BDS5. The LED's are listed below:

ACTIVE
SYS OK
CPU
FAULT
RELAY

- CURRENT LOOP COMPENSATION

The BDS5 has analog current loops. The current loop compensation components are all contained in the compensation module located on the front of the BDS5. The current loop compensation changes when you change the motor model. You must install the correct compensation module when changing motor models.



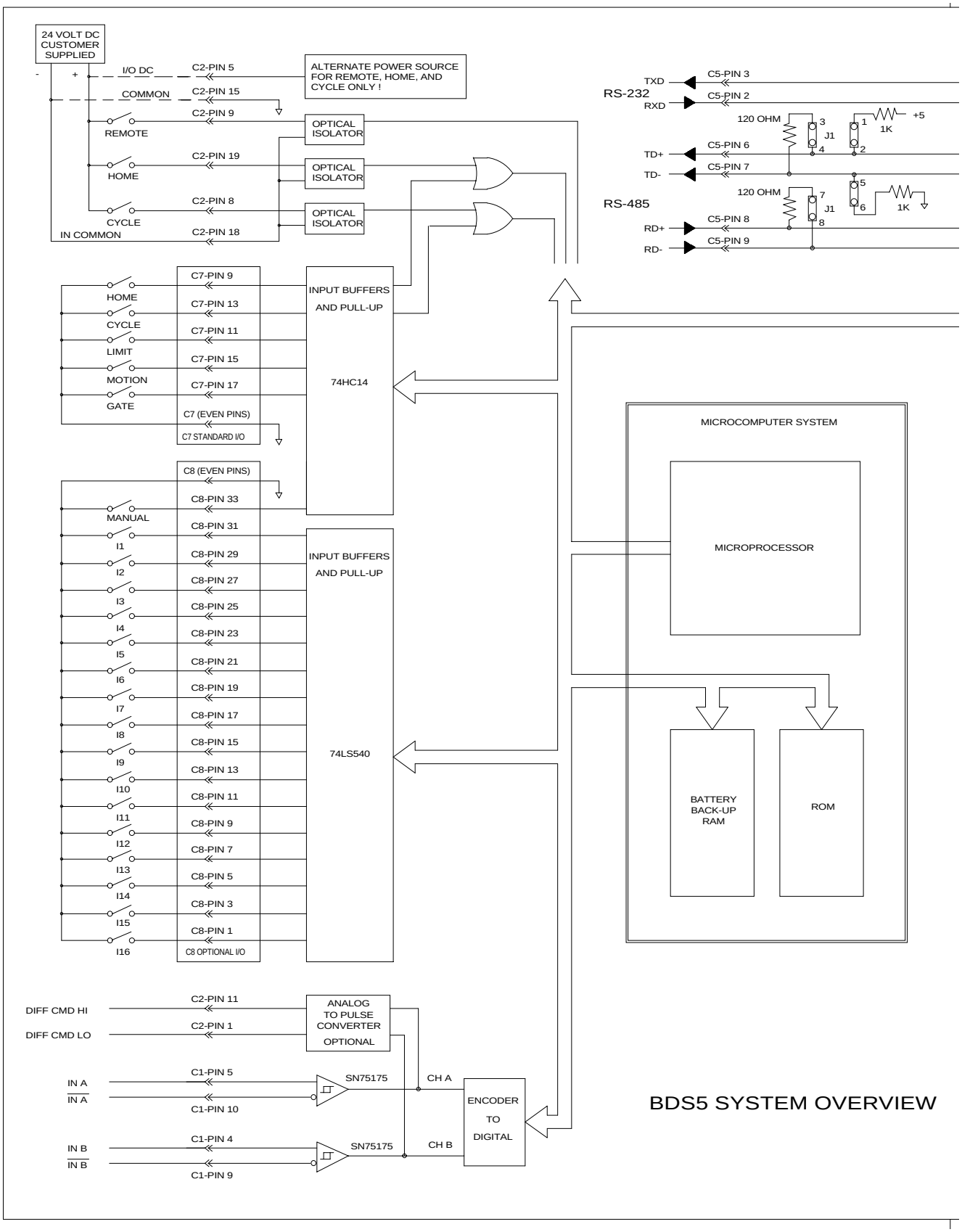
CAUTION

YOU MUST HAVE THE PROPER COMPENSATION MODULE INSTALLED FOR YOUR MOTOR

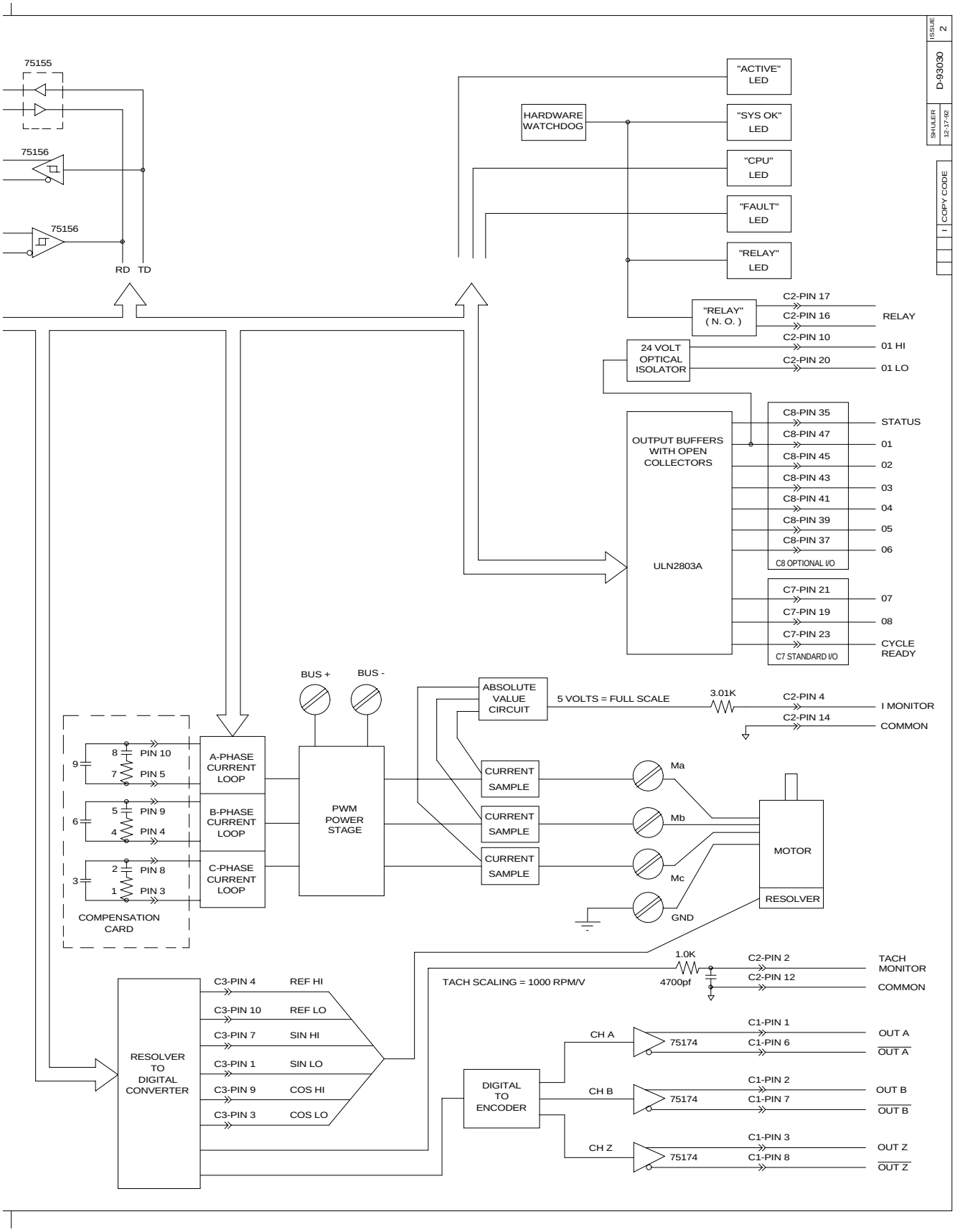
Failure to install the proper compensation module can cause damage to the BDS5, the motor, or both.

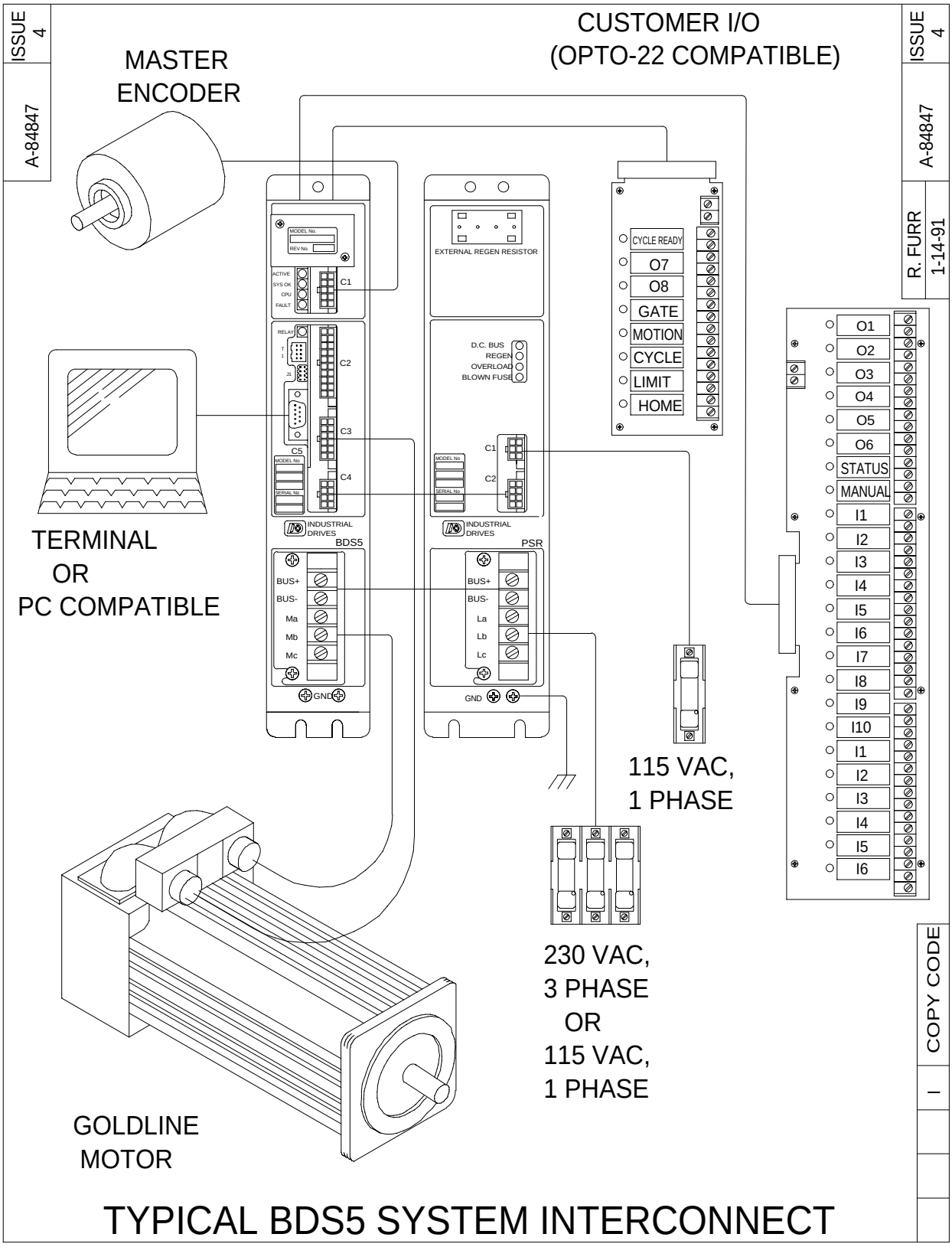
1.7 SIMPLIFIED SCHEMATIC DIAGRAM AND SYSTEM DIAGRAM

Drawings D-93030 and A-84847 illustrate a BDS5 servo system with all of the major components.



BDS5 SYSTEM OVERVIEW


ISSUE 2
D-93030
SHULER 12-17-92
COPY CODE I
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
C2-PIN 10
01 HI
24 VOLT OPTICAL ISOLATOR
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
PIN 10
PIN 5
PIN 9
PIN 4
PIN 8
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
TACH MONITOR
C2-PIN 12
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
CH B
CH Z
75174
75174
75174
C1-PIN 1
C1-PIN 6
OUT A
OUT A
C1-PIN 2
C1-PIN 7
OUT B
OUT B
C1-PIN 3
C1-PIN 8
OUT Z
OUT Z


ISSUE 4
A-84847
CUSTOMER I/O
(OPTO-22 COMPATIBLE)
MASTER
ENCODER
EXTERNAL REGEN RESISTOR
CYCLE READY
O7
O8
GATE
MOTION
CYCLE
LIMIT
HOME
D.C. BUS
REGEN
OVERLOAD
BLOWN FUSE
O1
O2
O3
O4
O5
O6
STATUS
MANUAL
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I1
I2
I3
I4
I5
I6
C1
C2
C3
C4
C5
INDUSTRIAL DRIVES
BDS5
PSR
BUS+
BUS-
Ma
Mb
Mc
La
Lb
Lc
GND
TERMINAL
OR
PC COMPATIBLE
115 VAC,
1 PHASE
230 VAC,
3 PHASE
OR
115 VAC,
1 PHASE
GOLDLINE
MOTOR
R. FURR
1-14-91
COPY CODE
TYPICAL BDS5 SYSTEM INTERCONNECT

CHAPTER 2

GETTING STARTED

2.1 INTRODUCTION

The information in this chapter will enable you to get started with programming the BDS5. Computer requirements and software installation prepare you for Motion Link, the Industrial Drives' software package specially designed for the BDS5. This chapter also contains an overview of Motion Link and its basic functions and features. The Motion Link setup program is introduced to enable the user to have easy access to the more common Motion Link procedures.

2.2 COMPUTER REQUIREMENTS

The BDS5 requires an IBM-PC or compatible computer with the following features:

- IBM-PC, XT, AT, PS/2, or compatible workstation.
- 512 K RAM.
- PC-DOS or MS-DOS Version 2.5 or later.
- Either 5-1/4" or 3-1/2" Floppy Drive.
- Standard Video Adapter (CGA, MDA, EGA, MCGA, and VGA).
- Serial Port (for communication link with BDS5). The serial communications port may be COM1 or COM2. The chart below shows the way your PC should configure COM1 and COM2. This is the normal configuration:

  COM1: (PC Address 3F8h, Interrupt Request #4)

  COM2: (PC Address 2F8h, Interrupt Request #3)

2.3 SOFTWARE INSTALLATION

The following section will show you how to back up and copy the files from the Motion Link disk to your computer's hard disk or floppy disk.

2.3.1 Backing Up the Disk(s)

Before starting Motion Link, you should back up the Motion Link disk(s) that came with the BDS5. This way if something happens to the master disk(s), you'll always have a copy. Remember, disks can be damaged by heat, magnets, pressure, and dirt — all extensively found in a manufacturing environment. Follow the procedure below to back up your disk(s).

1. From DOS, find either the DOS disk or directory where DISKCOPY.COM is located

and type:

```
DISKCOPY A: A:
```

Press enter and follow the DOS prompts on screen concerning source (Motion Link) and destination (blank disk) disks.

2. After DOS finishes copying the disk(s), place the Motion Link original disk(s) in a safe place for storage. Use it only to make extra copies. Never use the original disk(s) in day-to-day operation.

2.3.2 Software Installation

Motion Link can be installed on either a hard disk, 5-1/4 floppy disks, or 3-1/2 floppy disks. Follow the corresponding instructions below for the installation that your system requires.

2.3.2.1 Install on a Hard Disk

Use this procedure to install Motion Link on a hard disk.

1. Type:

```
C:
```

2. Make a subdirectory named ML5 on your hard disk. Type:

```
MD \ML5
```

3. Change to subdirectory ML5. Type:

```
CD \ML5
```

4. Insert the Motion Link disk into the A-drive. This disk should be in the disk holder in the front of this manual.

5. Copy all the files from the Motion Link disk onto the hard disk by typing:

```
COPY A:*.*
```

6. Store the original Motion Link disk in a safe place. Do not use this disk, except to make other copies.

2.3.2.2 Install on a Floppy Disk

Use this procedure to install Motion Link on a floppy disk. Use the procedure for both Motion Link disks if you are using a 5-1/4 floppy.

1. Insert your DOS disk into the A-drive. The DOS Disk must have the DOS file, FORMAT.COM.

2. Insert a blank disk into the B-drive.

3. Type:

```
A:
```

4. Type:

```
FORMAT B:/S
```

5. The Format program will ask you to hit a key to continue.

6. After the format is completed, your computer will prompt you to format more disks; answer "N" to exit the Format command.

7. Remove the DOS disk from the your computer. Leave the formatted disk in the B-drive.

8. Insert the Motion Link disk into the A-drive. This disk should be in the disk holder in the front of this manual.

9. Copy all the files from the original Motion Link disk onto your disk by typing:

```
"COPY A:*.* B:".
```

10. Label your disk as Motion Link. Include today's date on the label.

11. Store the original Motion Link disk in a safe place. Do not use this disk, except to make other copies.

2.3.3 Establishing Communications

This procedure will get you started using Motion Link after you have installed it:

1. Connect and turn on your BDS5 as described in the *Installation and Setup Manual*.

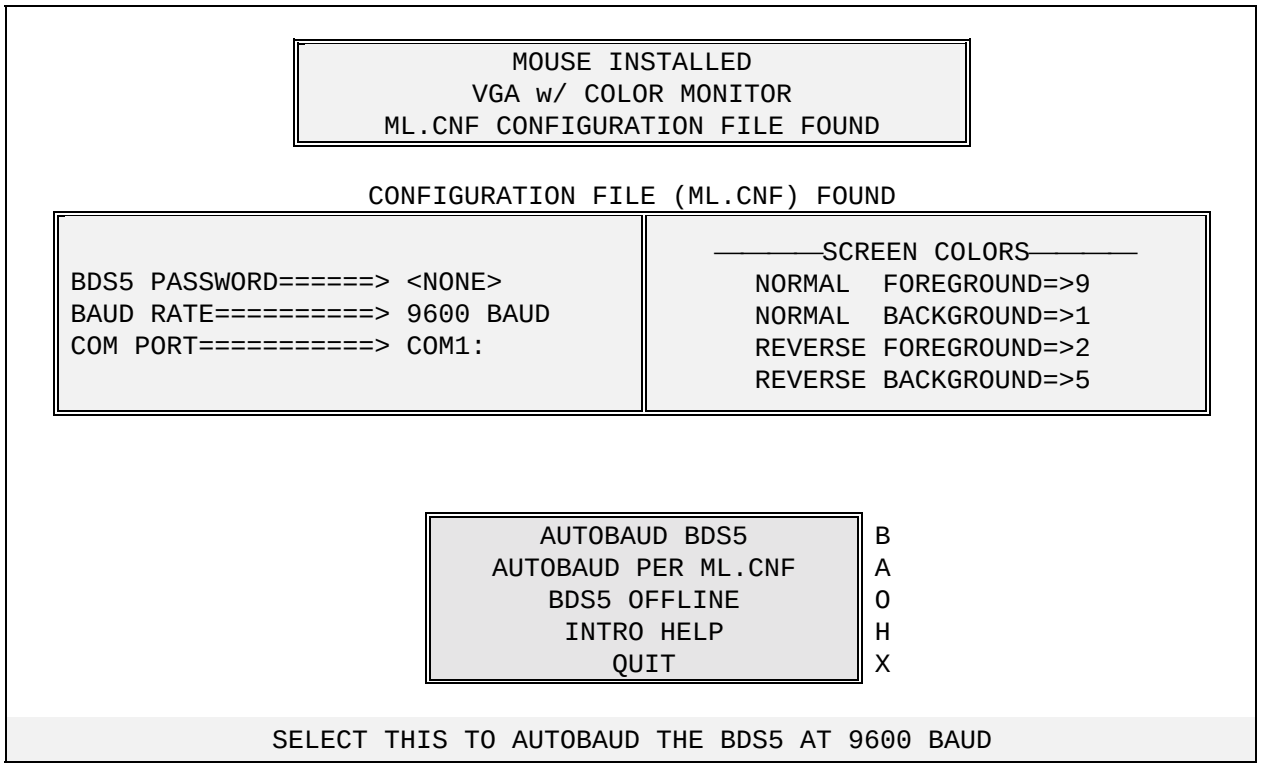


Figure 2.1. BDS5 Introduction Screen

2. If you have Motion Link installed on a hard disk, type:

```
CD\ML5
ML
```

Skip to Step 4.

3. If you have Motion Link installed on a floppy disk. Insert the copy of Motion Link in the A-drive, type:

```
A:
ML
```

4. When Motion Link responds the BDS5 should respond on your PC monitor with the message in Figure 2.1 (or a similar one).

This screen displays the current BDS5 configuration. The small box at the lower half of the screen provides five choices for the operator — Autobaud BDS5, Autobaud Per ML.CNF, BDS5 Offline, Intro. Help, and Quit. First time users may wish to refer to the online Intro Help by pressing "H." Choosing to auto baud with the BDS5 allows direct interactive communication with the BDS5. The BDS5 interactive prompt is "-->." This means the BDS5 is waiting and ready for a command. When you type, you are talking to the BDS5 just as you would with a terminal. For example, type:

```
P "HELLO, WORLD"
```

and the BDS5 should response by printing:

```
HELLO, WORLD
```

You can enter any BDS5 command from Motion Link just as if your IBM-PC compatible computer were a terminal.

The green SYS OK LED on the front of the BDS5 should turn on and remain on at all times after power-up (and autobauding).

2.4 MOTION LINK OVERVIEW

Motion Link is a full-featured communications program written by Industrial Drives and designed especially for the BDS5. Motion Link makes your IBM-PC compatible into a smart terminal. Of course, you can enter BDS5 commands from your computer as if you were using a terminal. For example, you can start a program with the RUN command and use the PRINT command to display values of variables. Motion Link also provides "smart" features such as a

full-screen editor, disk storage and retrieval, and the communications "capture" for debugging.

2.4.1 Menus and Windows

Motion Link's special features are accessed through a menu bar printed at the top of your PC screen. When you select an entry from the menu bar, a pull-down window appears, allowing you to select an item. Press the F10 key, the right arrow key, or the left arrow key, to display the menu bar. You can leave a window or the menu bar by pressing the escape key. There are six choices on the menu bar:

- PROGRAM - Edit new or old BDS5 programs; retrieve a program from disk or from the BDS5.

- VARIABLE - Edit new or old BDS5 variable files; retrieve variable files from disk or from the BDS5. Variable files contain static assignments. A static assignment is an instruction that sets the value of a variable that does not change throughout the program. Of course, static assignments can be included in your BDS5 power-up routine. However, moving a static assignment from your program to the variable file saves space in your program.

- CAPTURE - Start or stop capturing communications from the BDS5; retrieve previous capture files from disk; examine (edit) capture files.

- SCOPE - Retrieve, plot, print, and store PC-scope.

- OPTIONS - Set up communications, screen colors, computer configuration.
- HELP - Provide on-screen help for Motion Link and for the BDS5.

- UTILITIES - Exit Motion Link, enter a DOS command from within Motion Link, use a DEP01 Simulator or run Motion Link Setup program.

2.4.1.1 Program

The PROGRAM pull-down window allows you to retrieve, edit, transmit, and save BDS5 programs.

- EDIT - This selection calls the Motion Link Editor and assumes that you want to "re-edit" the last program that you edited. It is a short-cut, allowing you to edit without first loading a program from the BDS5 or from the disk. If you exit the Motion Link Editor, Motion Link remembers the program you were last editing. Note that if you have selected an item from either the VARIABLES or CAPTURE menu since you last edited a program, this selection is invalid.

- FROM DISK - This selection retrieves a program from your computer disk. Motion Link will display all of the files currently on your disk and allow you to choose the file you want. After you choose a program, the Motion Link Editor is called, allowing you to examine and change the program.

- FROM BDS5 - This selection retrieves the program currently stored in the BDS5. After the program is returned, the Motion Link Editor is called, allowing you to examine and change the program.

- NEW PROGRAM - This selection calls the Motion Link Editor, allowing you to enter a new program.

Upon exiting the Motion Link Editor, you can store the program to your computer disk and/or transmit it to the BDS5.

2.4.1.2 Variables

The VARIABLES pull-down window allows you to retrieve, edit, transmit, and save BDS5 variable files. A BDS5 variable file contains a list of some or all of the BDS5 variables with initial values. This includes user variables and control variables. Together, these variables configure a BDS5 for an application.

- EDIT - This selection calls the Motion Link Editor and assumes that you want to "re-edit" the last variable file that you edited. It is a short-cut, allowing you to edit without first loading a variable file from the BDS5 or from the disk. If you exit the Motion Link Editor, Motion Link remembers the variables you were

last editing. Note that if you have selected an item from either the PROGRAM or CAPTURE menu since you last edited a variable file, this selection is invalid.

- FROM DISK - This selection retrieves a variable file from your computer disk. Motion Link will display all of the variable files currently on your disk and allow you to choose the file you want. After you choose a variable file, the Motion Link Editor is called, allowing you to examine and change the variable file.

- FROM BDS5 - This selection retrieves all of the variables currently stored in the BDS5. After the variables are retrieved, the Motion Link Editor is called, allowing you to examine and change the variable file.

- NEW VARIABLES - This selection calls the Motion Link Editor, allowing you to enter a new set of variables.

Upon exiting the Motion Link Editor, you can store the variable settings to your computer disk and/or transmit them to the BDS5.

2.4.1.3 Capture



NOTE

This is a communications capture and is unrelated to the BDS5 variables CAP and CAPDIR which are for position capture.

- EDIT - This selection allows you to examine the communications that have been captured. Upon exiting the Motion Link Editor, you can store the captured data on your computer disk. Note that if you selected an item from either the PROGRAM or VARIABLES menu since you last captured communications or loaded a communications capture file, this selection is invalid.

- FROM DISK - This selection allows you to retrieve a capture file from disk and examine it with the Motion Link Editor.

- START CAPTURE - This selection starts (or re-starts) capturing communications from the BDS5. This selection always clears the capture storage area before beginning to capture new communications.

- STOP CAPTURE - This selection terminates the communications capture. If you want to examine the communications that were captured, select "EDIT" in this menu.

2.4.1.4 Scope

- VIEW AGAIN - This selection lets you view playback data that was previously retrieved from the BDS5.

- FROM DISK - This selection retrieves recorded data from your computer disk. Motion Link will display all of the playback files currently on your disk and allow you to choose the file you want. Playback files have the file type .CSV for "comma separated variables." This format is compatible with most spreadsheets.

- FROM BDS5 - This selection retrieves playback data stored in the BDS5. After the playback data is retrieved, the data is plotted and stored on disk.

- VIEW DATA - View the data in numerical (rather than graphical) format.

- PRINT PLOT - Print the plot on a line printer.

2.4.1.5 Options

- SELECT AXIS - This selection allows you to select options that are available to systems using RS-485 communications.

- BDS5 PASSWORD - This selection allows you to enter the password that you set in the BDS5 editor. If you set such a password in the BDS5, Motion Link needs the password to transmit new programs to the BDS5. If you use this selection to change the password, then you should use the UPDATE CONFIGURATION function below to write a new configuration file.

- COMMUNICATIONS - This selection allows you to set up your communications port. After you have set up this port, Motion Link will initiate an autobaud sequence to

re-establish communications. Remember to power-down the BDS5 so that it will autobaud. If you want Motion Link to use the new communications setup in the future, you must use the UPDATE CONFIGURATION function below to write a new configuration file on your computer disk.

- SCREEN COLORS - This selection allows you to change the colors displayed on your computer monitor. If you want Motion Link to use the new colors in the future, you must use the UPDATE CONFIGURATION function below to write a new configuration file.

- CABLE DISCONNECT - This selection provides a safe method of disconnecting the communication cable from a BDS5 that is powered up. After you have reconnected the cable, press the space bar and Motion Link will restart communications. Disconnecting this cable can generate random characters. Do not disconnect your communications cable without using this function.



NOTE

Always use this selection to secure data before disconnecting the communications cable.

- UPDATE CONFIGURATION - This selection allows you to examine and write the Motion Link configuration file. This file contains information about your computer, such as what communications port you are using, the baud rate at which your computer is transmitting, and what your screen colors are. All of the settings displayed in this selection can be changed by the SETUP-BDS5 PASSWORD, SETUP-COMMUNICATIONS, and SETUP-SCREEN COLORS and selections.

  After you make these changes, you should update the configuration file (ML.CNF) with this selection. This file is read by Motion Link when you type "ML" from DOS.

- TL FROM DISK - This selection is an internal function.

- TL FROM BDS5 - This selection is an internal function.

2.4.1.6 Help

- BDS5 HELP <F1> - This selection displays several pages of help for the BDS5. It lists BDS5 commands and variables with brief descriptions. You can also press F1 for this help.

- INTRO HELP - This selection displays introductory information about Motion Link.

- LAST COMMAND <F3> - This selection recalls your last command. You can also use the function key F3 to recall your last command.

- VARIABLE INPUT ^V - If you have included a variable input routine in your BDS5 program (that is, used VARIABLE$) and your program is running, this selection will initiate that routine. You can also press ^V (hold the control key down and press V) for this function.

- STOP MOTION ^X - This selection breaks your BDS5 program and stops motion. It works even if your program is not in the Interactive or Monitor mode. You can also press ^X for this function.

2.4.1.7 Utilities

- RUN DEP01 SIMULATOR - This selection allows the computer to simulate Industrial Drives DEP (Data Entry Panel).
- RUN BDS5 SETUP PROGRAM - This selection provides utilities to test I/O, drive feedback, communication, and dedicated switches. Refer to Section 2.6 for more information.

- EXIT TO DOS Alt-X - This selection terminates Motion Link and returns to DOS. You can also press Alt-X (hold the alternate key down and press X) for this function.

- SHELL TO DOS - This selection allows you to temporarily exit (or "shell") to DOS so that you can execute a DOS command. Type "EXIT" to return to Motion Link.

2.4.2 Editor

The Motion Link Editor is a full-featured screen editor. Use this editor to examine or edit programs and variable files, or to capture data. All of the editor commands can be accessed from a menu bar and pull-down windows. Press the F10 key to display the menu bar. Then use the left and right arrow keys to select a pull-down window. Each editor command can be accessed with a "control key" or "hot-key" sequence. You can use the control key as a shortcut in place of selecting from the window. The control-key sequence is listed beside each command here, and in Motion Link. For example, the FILE-PRINT selection can be accessed with ^P (hold the control key down and press P). Many selections require two control keys, such as FILE-FILE MERGE ^K^R. In this case, hold down the control key and press and release K, then press R. The rest of this section will discuss each of the editor pull-down windows.

2.4.2.1 File

- SAVE FILE ^K^S - Copy the file in the editor to the disk.
- MERGE FILE ^K^R - Copy a file into the editor starting at the cursor. You must place the editor cursor in the proper location before you make this selection.
- PRINT... ^P - Print the contents of the editor.
- EXIT <Esc> - This selection exits the Motion Link Editor. If you modify your program, Motion Link will prompt you to save your program to your computer's disk when you exit. If you are editing a program or a variable set, Motion Link will normally prompt you to transmit the program or variable settings to the BDS5. You can also use the escape key for this function.

2.4.2.2 Edit

- MARK START OF BLOCK ^K^B - This selection marks the beginning of a block. If you want to move or eliminate a block of text, use this selection to mark the top and the bottom of the block you want to manipulate.
- COPY MARKED BLOCK ^K^C - Use this selection after you have marked a block. This selection copies the marked block into the Motion Link cut/paste memory. If you want to copy the block into the cut/paste memory and delete it from the editor, see CUT MARKED BLOCK below.
- CUT MARKED BLOCK ^K^V - Use this selection after you have marked a block. This selection copies the marked block into the Motion Link cut/paste memory and deletes it from the editor. If you want to copy the block into the cut/paste memory without deleting it from the editor, see COPY MARKED BLOCK above.
- PASTE CUT/COPIED BLOCK ^K^P - Use this selection after you have either copied or cut a block to the cut/paste memory. This selection copies the cut/paste memory into the editor starting at the cursor. You must position the cursor to the proper place before you make this selection.
- SAVE MARKED BLOCK ^K^W - Use this selection after you have marked a block. This selection saves the marked block to a file on your disk. Motion Link will ask you for the file name after you make this selection.

2.4.2.3 GOTO

- FIND A STRING ^Q^F - This selection finds a string in the editor. Motion Link will prompt you to enter the string.
- REPEAT LAST FIND ^L - This selection repeats the last FIND A STRING.
- GOTO A LINE NUMBER ^Q^I - This selection moves the cursor to the specified line. Note that you can transmit your program to the BDS5 without comments. Since comment lines can be ignored by Motion Link when your program is transmitted, the line numbers of your program in the editor may not agree with the line numbers of your program in the BDS5. Because of this, Motion Link will ask you if you want to count comments. If you are trying to find a line number from a BDS5 error message, and you transmitted your

program without comments, specify that you DO NOT want Motion Link to count comment lines. Otherwise, specify that you DO want comment lines counted.

- SHOW SIZE OF EDITOR ^Q^O - This selection displays how much space is left in the Motion Link Editor. Use this selection if you are concerned that your program is filling up the editor. The Motion Link Editor can hold up to 2,000 lines and up to about 24,000 bytes.

- SHOW FREE MEMORY ^K^F - This selection displays how much space is left for your BDS5 program. Use this command if you are concerned that your program will fill up the BDS5 program memory.

2.4.2.4 Insert/Delete

- DELETE A WORD ^T - This selection deletes the next word after the cursor.

- DELETE TO END OF LINE ^Q^Y - This selection deletes from the cursor to the end of the line.

- DELETE A LINE ^Y - This selection deletes the entire line that the cursor is on.

- UNDELETE A LINE ^U - This selection inserts the last deleted line in the editor, starting at the cursor.

- INSERT A NEW LINE ^N - This selection inserts a blank line in the editor.

- DELETE ENTIRE EDITOR ^K^Y - This selection clears the entire Motion Link Editor.

2.4.2.5 Cursor

Table 2.1 shows the cursor control keys. Special keys are shown between greater than and less than symbols; for example, the Home key is shown as <Home>.

Table 2.1. Cursor Control Keys

TOP OF EDITOR	^<PageUp>
END OF EDITOR	^<PageDn>
UP ONE PAGE	<PageUp> or ^R
DOWN ONE PAGE	<PageDn> or ^C
BEGINNING OF LINE	<Home>
END OF LINE	<End>
UP ONE LINE	<Up> or ^E
DOWN ONE LINE	<Down> or ^X
LEFT ONE WORD	^<Left> or ^A
RIGHT ONE WORD	^<Right> or ^F
LEFT ONE CHARACTER	<Left> or ^S
RIGHT ONE CHARACTER	<Right> or ^D

2.4.2.6 Help

- BDS5 HELP <F1> - This selection displays several pages of help for the BDS5. It lists BDS5 commands and variables with brief descriptions. You can also press F1 for this help.

- THIS HELP SCREEN <F10> - Displays a help screen.

2.4.3 Types Of Data Files

Motion Link stores, retrieves, displays, and edits three types of data files. Each type has a different *file extension* or *file type*. *File extension* refers to the characters in the file name that follow the period. For example, the file TEST.BDS has the file extension "BDS." The three types of files are:

BDS Programs for the BDS5. Programs are also called *software*. Programs are transmitted to the BDS5 and can be run indefinitely.

VAR Variable sets for the BDS5. Variable sets are BDS5 variables that define an application. For example, you may have different variable sets to change the tuning when the application requires it. Variable files may include some or all of the BDS5 variables. For example,

your Motion Link disk has the file "STANDARD.VAR." This variable file includes all of the "standard" or "default" variable settings. Variable files are transmitted to the BDS5 to initialize variables before programs are run.

CAP Capture files contain captured communications from the BDS5. The capture features of the BDS5 allow you to collect and store up to 16,000 bytes of transmissions from the BDS5. Capture is provided to help you debug your program.

Any time you store your programs, variable sets, or captured communication onto your computer disk, Motion Link will automatically determine the proper file extension.

2.4.4 Using IBM-PC Compatibles

If you are using an IBM-PC Compatible, make sure it has been booted with the optional serial/parallel adapter plugged in. Also make sure it has been configured to allow the operation of the serial/parallel adapter on batteries. This configuration may be accomplished through the IBM-supplied program SYSPROF.COM. To run this program, type "SYSPROF"<cr>.

This program may also be reached through the APP/SELECTOR DISK, but this is a remain-resident program that will not leave enough memory to load Motion Link after running. So a three-key computer reboot (Ctrl, Alt, Del) must be done to remove this program from memory before loading Motion Link. Because of this, it is easier to simply run "SYSPROF" in order to configure the IBM CONVERTIBLE.

2.5 MOTION LINK SETUP PROGRAM

The Motion Link Setup Program is accessed through the Utilities Menu. Setup provides the following test capabilities:

- Communicate with the BDS5
- Resolver Zeroing Test
- Tune Drive
- Drive Test
- Drive Feedback
- Input Test
- Output Test
- Machine Setup - Units
- Machine Setup - Limits
- Motor Setup
- BDS5 Modes
- Communications
- Other
- Send Variables
- Reset Variables

This test program provides the operator with user friendly methods for testing most BDS5 functions.

2.6 PROCESSOR MODES

2.6.1 Prompts

The BDS5 provides several modes of operation. Each mode is distinguished by a unique prompt. A prompt is the short series of characters that the BDS5 writes to the screen asking you for input. For example, the interactive prompt is "-->." This prompt is unique and tells you that the BDS5 is in the Interactive mode.

The BDS5 is designed to receive commands from a terminal or a computer through a serial port. In order to support computer communications, the BDS5 observes the following conventions:

Table 2.2. BDS5 Rules For Prompts

1. Prompts are 3 characters long (except single-step and trace).
2. Prompts end with a greater than (“>”).
3. Each mode has a unique prompt.
4. Once the BDS5 displays a prompt, it stops transmitting until a new instruction and/or a carriage return is received.

These conventions are designed to allow efficient communications between the BDS5 and a computer. The last rule ensures that there is never a question about which device is transmitting. If a ">" has been issued from the BDS5, then the BDS5 will not transmit anything until a carriage return or escape has

been entered. The only exception is if you program the BDS5 to print a ">" from a PRINT or INPUT command. The BDS5 will allow ">" in print statements, though this is considered a poor practice if you are using a computer to communicate with the BDS5.

Similarly, the BDS5 will not accept input unless a ">" has been issued by the BDS5. The INPUT command is the only exception to this rule. This rule can be awkward if you are using the BDS5 from a terminal; if an error occurs during the interactive or monitor modes after the ">" has been displayed, the BDS5 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 the ">." This is because the trace prompt does not indicate that the BDS5 is waiting for input. If the BDS5 is communicating within a multidrop communication line, then the prompt is modified to include a prefix which indicates the axis address. The table below shows the prompts in both the normal (non-multidrop) and multidrop configurations. Note that the multidrop address is 65 or ASCII A.

Table 2.3. BDS5 Prompts

Mode	Non-multidrop (ADDR=0)	Multidrop (ADDR = 65)
Interactive	`-->`	`A->`
Monitor	`==>`	`A=>`
Single-step	`s-->`	`As->`
Trace	`t...`	`At..`
Edit	`e->`	`Ae>`
Load	l->	Al>
Edit/Insert	`i->`	`Ai>`
Edit/Find	`f->`	`Af>`
Edit/Change	`c->`	`Ac>`

2.6.2 Descriptions of Modes

The following section describes each of the modes of operation. Refer Figure 2.2 which is a diagram showing each mode and how it interacts with the other modes.

2.6.2.1 Interactive Mode

The BDS5 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 prompt (-->) is written to the screen, and the BDS5 awaits a new command. Your program is not running if the BDS5 is in the Interactive mode.

Refer to Figure 2.2. There are many ways to enter the Interactive mode. First, if the power-up label (POWER-UP$) is not present, the BDS5 will power-up in the Interactive mode. The BREAK (B) command and errors that break program execution cause the BDS5 to exit the Run mode and enter the Interactive mode.

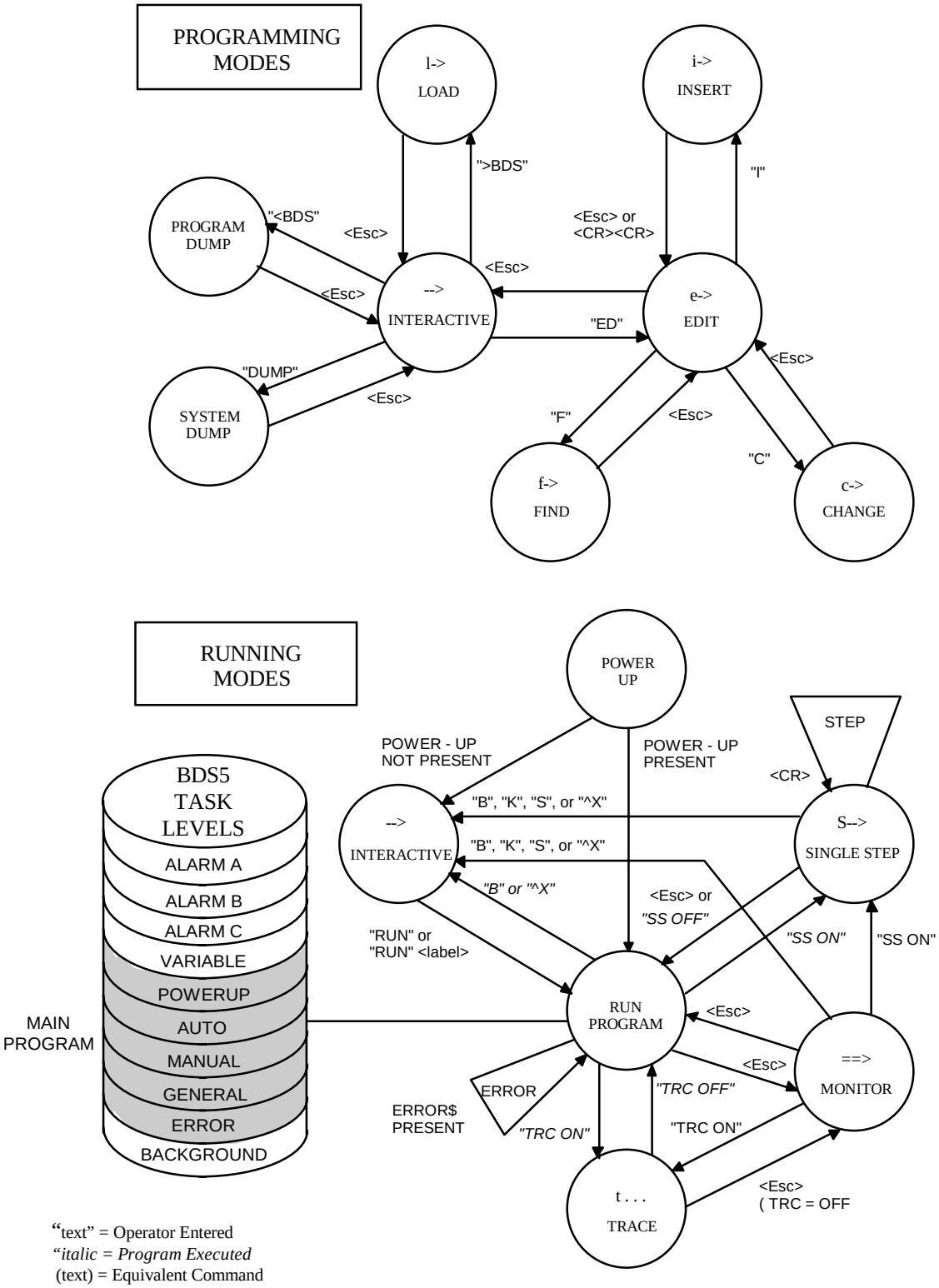


Figure 2.2. BDS5 State Table

2.6.2.2 Run Mode

The BDS5 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 2.2. 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 BDS5 will start running your program at that label on power-up. Also, the BDS5 will exit the Run mode to the Monitor mode if the escape key is pressed.

Errors can also cause the BDS5 to change modes. Some errors are serious enough to cause the BDS5 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.

2.6.2.3 Monitor Mode

The BDS5 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 any variable and change any programmable variable from the Monitor mode. The commands that are allowed from the Monitor mode are a subset of the commands allowed from the user program and Interactive modes, and include the following commands:

Table 2.4. 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 BDS5 to the Interactive mode.

2.6.2.4 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 BDS5 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 BDS5 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 BDS5 to the Run mode.

2.6.2.5 Trace Mode

The Trace mode is provided for debugging. When in trace, the BDS5 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 prompt is not a true prompt in that you are not allowed to enter a command after the prompt is issued. This is why it does not have the ">" that the other prompts use to indicate that the BDS5 is waiting for a command.

The trace is enabled by turning TRC on. When TRC is on, the BDS5 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 exit to the Monitor mode and turn TRC off. If TRC is turned off from the program, the BDS5 will exit to the Run mode. If both TRC and SS are on, then the BDS5 will be in Single-Step mode.

2.6.2.6 Other Modes

The other modes shown in Figure 2.2 include the Edit modes (Edit, Insert, Change, and Find) and the communication modes (Program Load, Program Dump, and System Dump). These modes are covered in later chapters.

CHAPTER 3

PROGRAMMING LANGUAGE

3.1 INTRODUCTION

This chapter discusses the basics of the BDS5 and its programming language.

Your BDS5 system should be mounted and wired as described in the *Installation and Setup Manual*. The AC Line voltage to your PSR4/5 should not be turned on for examples in this chapter. Turn on Control Power only, and establish communications. If the proper connections are not made, or the terminal is not communicating, then see the *Installation and Setup Manual*.



WARNING

AC LINE SHOULD NOT BE TURNED ON!

3.2 INSTRUCTIONS

The BDS5 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.

```
J 10
```

would cause the motor to jog at 10 RPM.

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

3.2.1 Comments

Instructions can be followed by comments on the same line. A semicolon marks the beginning of a comment. The BDS5 ignores everything on the line after the semicolon. For example:

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

is a valid instruction. The BDS5 ignores everything that follows the semicolon. 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
```

3.3 VARIABLES

The BDS5 uses variables to monitor and control virtually all of its processes.

3.3.1 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 E lists variable with its units. Units are programmable; when shipped from the factory the standard settings are as follows:

Table 3.1. 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 BDS5. 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 BDS5 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 BDS5 is driving a lead screw, velocity could be programmed in inches/minute. If you want to change the units, see Chapter 4. Examples in this manual will assume that the BDS5 is configured with standard units.

3.3.2 Three Types of Variables

The BDS5 has many variables, all of which are listed in Appendix E. 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 BDS5 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 BDS5. An example of a control variable is current limit, ILIM. ILIM limits the maximum current the BDS5 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.

3.3.3 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 E.

3.3.4 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.

3.3.5 Printing Variables

All variables can be displayed. To display a variable on the terminal, you should use P, the PRINT command. For example, type:

```
P 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 BDS5, rotate the motor shaft about half a revolution. Now, print PFB as above and notice that it has changed to reflect the new position.

3.3.6 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 '='
```



NOTE

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) and type:

```
ILIM=100
```

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

3.3.7 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 BDS5 is disabled. Attempting to change AMAX with the BDS5 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 BDS5 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 E.



NOTE

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

3.3.8 Power-up and Control Variables

Most control variables and all user variables are stored in non-volatile RAM; their values are not lost when the BDS5 is powered-down. In general, control variables are remembered, except the switches. Table 3.2 shows the condition of all BDS5 programmable switches on power-up.

Table 3.2. Power-Up State of Programmable Switches

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

The output word, OUT, is set to zero shortly after power-up.

3.3.9 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.

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

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

DIR Sets BDS5 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 BDS5. 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 the *Installation and Setup Manual* 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. 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.

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.

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 set to 1 on power-up and left at 1 for preliminary operation.

PLIM — Enable software travel limits. This variable is set to 1 on power-up. This variable is normally set to 0 during preliminary operation.

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 VMAX_1.2. 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 BDS5 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..X250	User variables. Initially set to 0.
XS1..XS50	User switches. Initially set to 0.
ZERO	Puts the BDS5 in Resolver Zeroing mode. This is set to 0 on power-up. Zeroing mode is used only during installation. If 1, BDS5 rotates the motor to the zero position. If 0, the BDS5 controls the motor normally.

3.3.10 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 250 user variables: X1, X2, . . . X250. (Extended to 750 with Varaable EXTDX=1. PC Scope is not available with EXTDX=1). 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.

3.3.10.1 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 BDS5 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 BDS5 responds:

```
1000
```

since X(X10) is now X2, which equals 1000. So printing X10 indirect, X(X10), prints the user variable to 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.

3.3.11 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 BDS5 should respond by printing 1.

3.3.12 Special Constants

The examples above have used decimal numbers in most of the assignments. There are four special constants that make the BDS5 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 (that is, 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 "=" is not used with Y, N, ON, and OFF.

3.4 MATH

3.4.1 Hexadecimal

The BDS5 allows constants to be entered in hexadecimal, or hex. Hex is base 16 representation

which is often used when programming computers. BDS5 hex constants begin with a number and are followed by an "h." For example: 16h, 0Fh and 0FFh are all hex numbers. Appendix H shows the hex conversion of 0 through 255. From the appendix, you can see that hex 25 is equal to decimal 37. The two instructions:

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

have identical effects because 25 hex equals 37 decimal. 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.

3.4.2 Algebraic Functions

The BDS5 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 3.3.

Table 3.3. 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 expression:

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

fills X3 with 600.

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.

3.4.3 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 BDS5 responds: 3,

since

```
        00000001 (Binary 1)
OR      00000010 (Binary 2)
        --------
        00000011 (Binary 3)
```

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

The BDS5 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.

3.5 GENERAL PURPOSE INPUT/ OUTPUT

The BDS5 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
```

and either 1 or 0 will be displayed.

3.5.1 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 3.4.

Table 3.4. 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 3.5. 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

3.6 FAULT LOGIC

This section covers how to enable the BDS5 and how faults affect the operation. This discussion will center around Figure 3.1. This drawing has six areas, each of which is labeled with an encircled 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 BDS5 system should be mounted and wired as described in the *Installation and Setup Manual*. The AC Line to your PSR4/5 should not be turned on initially for examples in this chapter. If the proper connections are not made, or the terminal is not communicating, then see the *Installation and Setup Manual*.



WARNING

AC LINE SHOULD NOT BE TURNED ON.

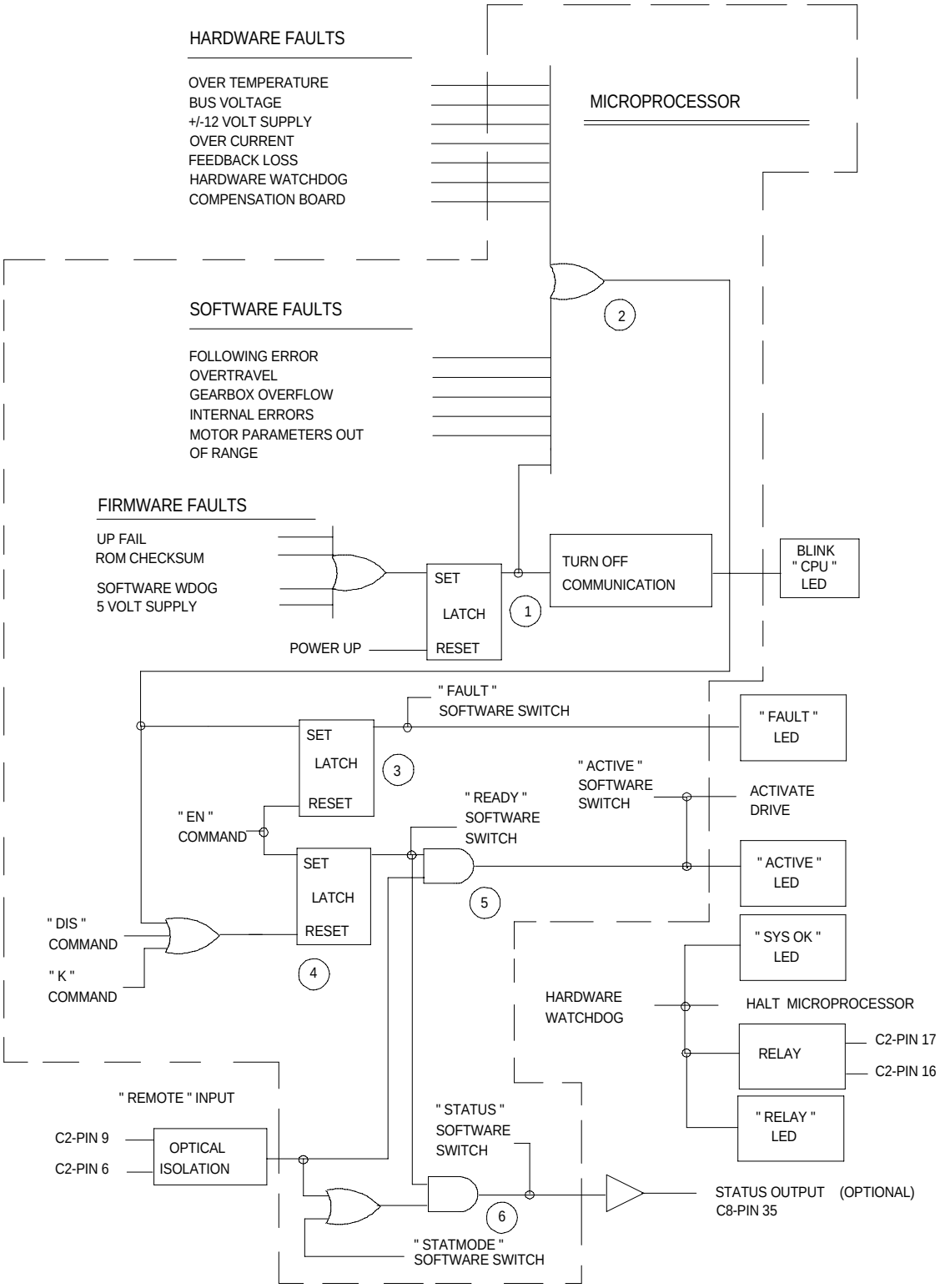


Figure 3.1. BDS5 Enable/Fault Logic Diagram

3.6.1 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 3.1, 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 D. The only way to reset these faults is to power-down the BDS5. 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.

3.6.2 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 BDS5, as described in Appendix D.

3.6.3 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 BDS5.

3.6.4 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.

3.6.5 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 BDS5 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 BDS5. If you cannot turn REMOTE on, see the *Installation and Setup Manual*. Note that some faults "hide" the value of the REMOTE input from the BDS5 microprocessor. This does not normally matter because all faults must be cleared before the drive will enable. If this condition exists, the BDS5 will print REMOTE as "-1."

3.6.6 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. STATUS can be used for an interlock. In this case, you want STATUS to indicate drive ACTIVE. If the BDS5 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 BDS5 is ready for the REMOTE input to turn on. That is, if REMOTE turns on, the BDS5 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.



WARNING

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

3.6.7 Motor Brake

Industrial Drives 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 the BDS5 is disabled or powered down, the brake will be active.

3.6.8 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 BDS5 passes its power-up self tests. Then the contacts close and the BDS5 begins normal operation. Note that if the BDS5 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 BDS5 from operation and contact the factory.

3.7 DRIVE CONTROL

This section discusses several variables that you must be familiar with before you can control the BDS5.

3.7.1 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.

3.7.2 Position

3.7.2.1 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 3.6. PCMD is set to PFB when the BDS5 is disabled.

PFB, the position feedback, is the actual position of the motor. It is updated every millisecond. PFB is in position units. Section 3.3.5 explained how to look at PFB and watch it as the motor turns. PFB is always active, even when the BDS5 is disabled. PFB is reset to zero when the BDS5 is powered-up.

3.7.2.2 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 BDS5 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 BDS5 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.

3.7.2.3 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:

Table 3.6. 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

The BDS5 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.

3.7.2.4 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. This is because the BDS5 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:

```
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.

3.7.3 Velocity

3.7.3.1 VCMD, VFB, VE, & VAVG

VCMD is the commanded velocity. Like PCMD, VCMD is generated internally from motion commands. VCMD is zero when the BDS5 is disabled. VCMD is in velocity units.

VFB is the feedback velocity. It is updated every millisecond. VFB is always active, even when the BDS5 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 (that is, 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.

3.7.3.2 Velocity Limits, VMAX & VOSPD

VMAX is the BDS5 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 7500 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. VMAX is in velocity units.

VOSPD is the maximum velocity for your system. The BDS5 generates an overspeed fault if VFB is ever greater than VOSPD. You can set VOSPD to any level below 1.2_VMAX. This allows you to limit the speed of your system to any level below VMAX. When an overspeed occurs, the BDS5 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 BDS5 is disabled. VOSPD is in velocity units.

3.7.4 Current

3.7.4.1 Motor Current, ICMD & 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.

3.7.4.2 Current Limits, IMAX & ILIM

IMAX is the maximum level of current that the BDS5 can output. It is set at the factory; its value depends on both the BDS5 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 BDS5 can output. You can set ILIM at any time, even during profile moves. ILIM is in current units.

3.7.5 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 BDS5 is running as a velocity loop. Most positioning applications run with PL on. See later in this chapter for more information about the position loops. PL turns on at power-up. You can change PL at any time.

3.7.6 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 later in this chapter for more information.

3.7.7 Enabling the BDS5



WARNING

THE BDS5 WILL BE ENABLED AND THE MOTOR WILL TURN. SECURE THE MOTOR.

At this point you should turn REMOTE on as described in the *Installation and Setup Manual*. 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 the *Installation and Setup Manual*.

THE MOTOR MAY MOVE UNEXPECTEDLY!

BE PREPARED TO DISABLE THE BDS5!



WARNING

You should have completed "Initial Check-Out" in the *Installation and Setup Manual*. If not, return to the *Installation and Setup Manual* and complete that section.

This section will enable the BDS5. The system may be unstable. The motor may begin oscillating or run away. Be prepared to disable the BDS5 quickly. You can disable the BDS5 by turning off (opening the contacts of) LIMIT or REMOTE.

To enable the BDS5, turn on the AC Line and enter the enable command:

```
EN
```

The BDS5 should turn on. To verify that it did turn on, print ACTIVE. If ACTIVE is 1, then the BDS5 is enabled; otherwise, it is disabled.

To disable the BDS5, enter the disable command:

```
DIS
```

As an alternative, you can disable the BDS5 with the one-letter kill command by typing:

```
K
```

ENABLE, DISABLE, and KILL are examples of BDS5 commands. All of the BDS5 commands are listed, with their formats and syntax, in Appendix C.



NOTE

Appendix C is a quick reference for all BDS5 commands.

3.7.8 Limiting Motor Current

The following section discusses how the BDS5 limits motor current.

3.7.8.1 Continuous Current, ICONT

The BDS5 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 BDS5 rating and on the motor; ICONT is set at the factory, and it is in current units.

Most BDS5 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 BDS5 from overheating.

3.7.8.2 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 (the continuous current rating of the BDS5), IMON (the current monitor), and time.

When the BDS5 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 BDS5 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.

3.7.8.3 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.

3.8 MOTION COMMANDS

This section discusses how to control motion using the BDS5. Basic motion commands are described first. Later sections discuss advanced motion control including BDS5 Macro Moves, electronic gearbox, and synchronizing motion.

3.8.1 Basic Motion Commands

3.8.1.1 AMAX, ACC, & DEC

The BDS5 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 which 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 BDS5 is disabled.



WARNING

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.

3.8.1.2 EN, STOP, & LIMITS

Before any motion can take place, the BDS5 must be enabled. Type:

```
EN
```

3.8.1.3 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 BDS5 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 the *Installation and Setup Manual* 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 BDS5 functions to operate properly.



WARNING

Do not use MOTION or any other BDS5 input for Emergency Stop. When Emergency Stop is activated, it should directly remove power from the system.

3.8.1.4 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 BDS5.

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.



NOTE

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 D lists all errors and their severity.

3.8.1.5 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.

3.8.2 Limiting Motion

The BDS5 allows you to limit motion with both Software and Hardware Travel Limits.

3.8.2.1 Hardware Travel Limits

Hardware Travel Limits limit the range of motion. 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 BDS5 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 BDS5 rather than just stopping motion, as the software limits do. This means that the motor must be backed away from the limit by hand.

The *Installation and Setup Manual* 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 BDS5 to be enabled. If the contacts open, the BDS5 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.

3.8.2.2 Software Travel Limits, PMAX & PMIN

Software Travel Limits limit the range of motion of the motor. 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 turned on at power-up. 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.

3.8.2.3 User Position Trip Points, PTRIP1 & PTRIP2

The BDS5 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 4.

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 BDS5 is powered-up, trip points are enabled.

3.8.3 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.

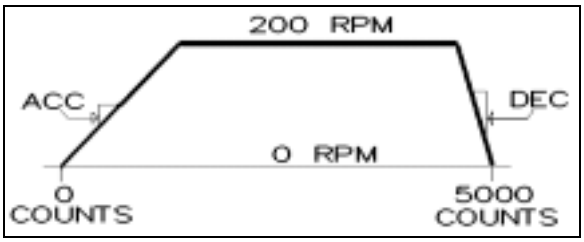


Figure 3.2. A Simple Profile

The graph in Figure 3.2 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.

3.8.3.1 S-Curves

The BDS5 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 3.3 shows the profile from Figure 3.2 using S-curves instead of straight lines.

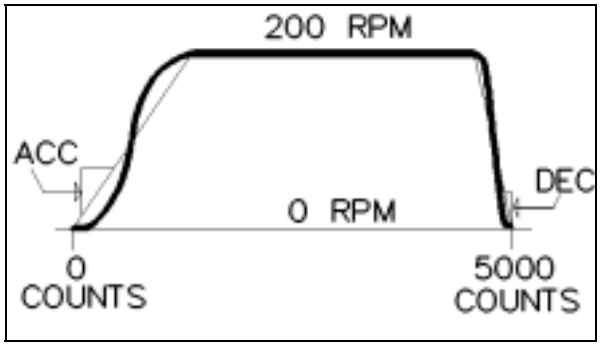


Figure 3.3. S-Curve Profile

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.

NOTE

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 BDS5 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 Industrial Drives application note B101, "Acceleration Profiles."

Table 3.7. S-Curve Acceleration Chart

For this acceleration...	Set SCRV to...
Straight-Line	1
Modified Polynomial	2
Polynomial	3
Modified Sinusoid	4
Sinusoid	5

3.8.3.2 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.
For example:

```
MA  50000  1000
```

moves to position 50000 at a peak speed of 1000 RPM.

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. For example,

```
MA  100000
```

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 F lists the execution times of a few simple moves.

NOTE

Not specifying the speed in MA commands reduces execution time.

3.8.3.3 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:

```
MI  5000  200
```

causes the motor to move 5000 counts at a peak speed of 200 RPM. 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.



NOTE

Not specifying the speed in MI commands reduces execution time.

3.8.3.4 Incremental Move Example



WARNING

SHOCK HAZARD!

Large voltages from the AC line and the DC bus can cause injury. Wire the BDS5 as described in the *Installation and Setup Manual*.

THE MOTOR MAY MOVE UNEXPECTEDLY!

BE PREPARED TO DISABLE THE BDS5!



WARNING

You should have completed "Initial Check-Out" in the *Installation and Setup Manual*. If not, return to the *Installation and Setup Manual* and complete that section.

This section will enable the BDS5. The system may be unstable. The motor may begin oscillating or run away. Be prepared to disable the BDS5 quickly. You can disable the BDS5 by 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. Now, to bring the motor back to the original position, type:

```
MI  -8000
```

3.8.3.5 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 BDS5 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 BDS5 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.

3.8.3.6 Multiple Profile Commands

The BDS5 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 BDS5 automatically pauses before calculating a third motion profile, thus stopping this error from occurring.

3.8.3.7 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
```

3.8.4 JOG (J) Command

This section describes J, the JOG command. Jogging is useful when you want to command motion without position endpoints. For example,

```
J  500
```

causes the motor to rotate at 500 RPM indefinitely. 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 directions 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.

3.8.5 NORMALIZE (NORM) Command

NORM, the NORMALIZE command, is required if you want to reset the BDS5 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 NORMALIZE command. For example,

```
NORM  10000
```

sets PFB (position feedback) as well as PCMD (POSITION command) to 10000 in position units. 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.

3.8.6 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 BDS5 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 later in this chapter.

3.8.7 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.

3.8.7.1 MCA, MCI, MCD, & 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 BDS5 uses the larger speed, either the beginning or ending speed, for the traverse speed. All velocities are specified greater than zero. The BDS5 determines the direction based on the specified position. If no velocities are specified, then the BDS5 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. For example,

```
MCD  100                ;100 MSEC DWELL
```

This example specifies a 100 millisecond 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 BDS5 resets the Macro move memory. If you are typing in a Macro move and you make an error, you should disable, then enable the BDS5, 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.

3.8.7.2 Macro Move Example #1

As an example of Macro moves, consider the following profile:

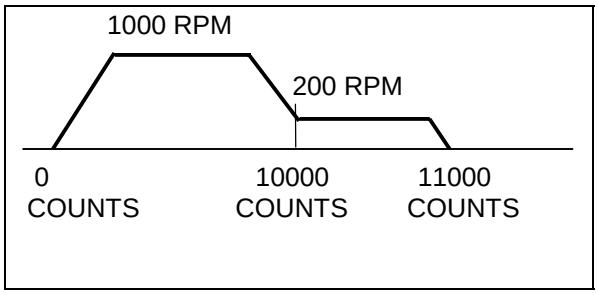


Figure 3.4. Macro Move Example #1

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 3.4.

```
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.

3.8.7.3 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 below:

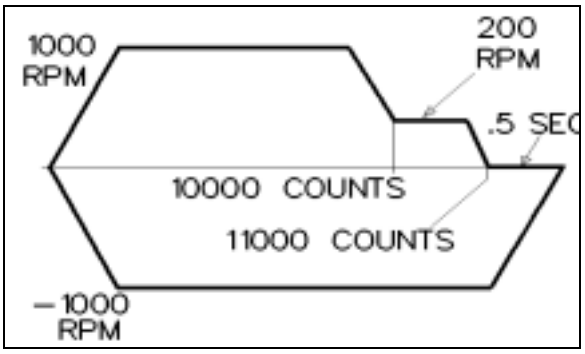


Figure 3.5. Macro Move Example #2

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.

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.

3.8.8 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 command:

```
MRD  1000  100
```

moves the motor so that the R/D output is 1000. 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. For example:

```
MRD  1000  100  CW  ;MOVE R/D TO 1000,
                    ;BUT ALWAYS CW
```

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 3.8.

Table 3.8 R/D Converter Accuracy

Resolution	Maximum Position
12	4095
14	16383
16	65535

MRD moves are not buffered. They are not allowed when the BDS5 is jogging or if a move is in progress.

MRD moves can be used to improve the accuracy of homing sequences. First, use the BDS5 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.

3.8.9 Capturing Position

Position capture is a feature where the position feedback (PFB) is *captured* when a hardware input transitions. The BDS5 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.

3.8.9.1 Enabling Capture, CAP & PCAP

The switch CAP controls capture. If CAP is on, then capturing is enabled. When capturing is enabled, the BDS5 will watch the HOME input. When the HOME input changes to the state specified by CAPDIR, the BDS5 will store the position in the variable PCAP. After the capture, the BDS5 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.

3.8.9.2 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.

3.8.9.3 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.

3.8.10 Clamping

Clamping stops BDS5 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 BDS5 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 BDS5 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.
```

3.8.10.1 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 BDS5 will run the machine into a stop and limit current to ILIM. When SEG is equal to 0, the BDS5

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 BDS5 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 BDS5 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.

3.8.11 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
```

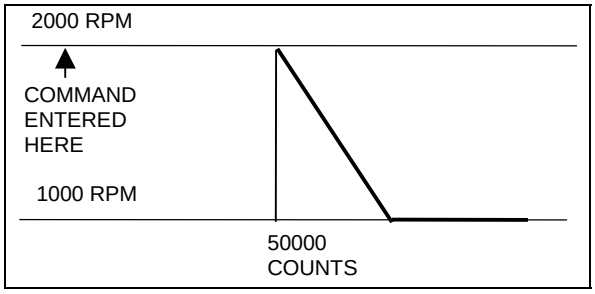


Figure 3.6. Jog From (JF) Command

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
```

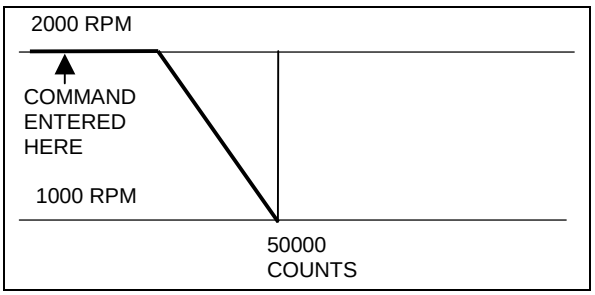


Figure 3.7. Jog To (JT) Command

Position dependent commands must be used with care. If you specify a position that has already passed, the BDS5 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 BDS5 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.

3.8.11.1 Registration

The BDS5 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 BDS5, 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).

3.8.11.2 Registration Example

The following example shows how to program the BDS5 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 BDS5 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 Industrial Drives application note "Cut to Length."

3.8.11.3 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 BDS5
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 20,000 counts) and finally come to rest at 30,000 counts. Actually, the motor will jog to about 40 RPM and continue at that speed until it comes to rest at 30,000 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 BDS5
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 4. For our example, the first TIL command can be replaced with "W 2" and the second can be replaced with "W 3."

3.9.11.4 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.

3.8.12 External Inputs

External inputs are normally from a "master" motor. As a standard, these inputs are in digital encoder format. Examples of "master" motors include the encoder-like output from another BDS5, output from an actual encoder, or a customer synthesized encoder signal. The external input can control motion in the two BDS5 Master/Slave modes: electronic gearbox and profile regulation. The BDS5, 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 BDS5.

External inputs are connected to Channel A and Channel 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 4 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.

3.8.12.1 Analog Input

The standard BDS5 is configured with digital external inputs. As an option, the BDS5 provides an analog external input. Note, however, that you

cannot have both types of inputs at the same time. For systems configured with analog inputs, the BDS5 converts the analog input to a pulse train, where 10 volts of input is equivalent to 2 MHz. If the analog input is a velocity command, then use electronic gearbox Master/Slave mode to make the BDS5 a velocity drive. See Appendix G for more information. If the analog input is going to be used for "feedrate override," use profile regulation.

The analog external input is connected to the analog input of the Customer I/O Connector.

3.8.13 Electronic Gearbox

Electronic gearbox is one of two BDS5 Master/Slave modes. Refer to Figure 3.8 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.

3.8.13.1 Gear Ratio, GEARI & GEARO

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:

$\text{REV}_{\text{MASTER}}$ is an arbitrary number of revolutions of the master motor,

$\text{REV}_{\text{SLAVE}}$ is the corresponding number of revolutions of the slave motor,

$\text{RESOLUTION}_{\text{SLAVE}}$ is the resolution of the slave motor in counts/revolution, and

$\text{RESOLUTION}_{\text{MASTER}}$ 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:

$$\frac{\text{GEARI}}{\text{GEARO}} = \frac{\text{REV}_{\text{SLAVE}} \times \text{RESOLUTION}_{\text{SLAVE}}}{\text{COUNTS}_{\text{MASTER}}}$$

where:

$\text{COUNTS}_{\text{MASTER}}$ is an arbitrary number of counts of the master signal and

$\text{REV}_{\text{SLAVE}}$ and $\text{RESOLUTION}_{\text{SLAVE}}$ are as before.

To enable the Gearbox mode, type:

```
GEAR ON
```

If the ratio is not an integer, the BDS5 does not "drop pulses." The BDS5 keeps track of partial pulses to eliminate dropping pulses over time. If the number of pulses coming into the BDS5 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.

3.8.13.2 Gearbox Example 1

Two BDS5'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?

$$\frac{\text{GEARI}}{\text{GEARO}} = \frac{\text{REV}_{\text{SLAVE}}}{\text{REV}_{\text{MASTER}}} \times \frac{\text{RESOLUTION}_{\text{SLAVE}}}{\text{RESOLUTION}_{\text{MASTER}}}$$

$$\frac{\text{GEARI}}{\text{GEARO}} = \left(\frac{1}{3}\right) \times \left(\frac{1}{1}\right) = \frac{1}{3}$$

You can select any integer values for GEARI and GEARO that have the ratio 1/3.

3.8.13.3 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 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.

3.8.13.4 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 BDS5 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:

BDS5 Master/Slaving

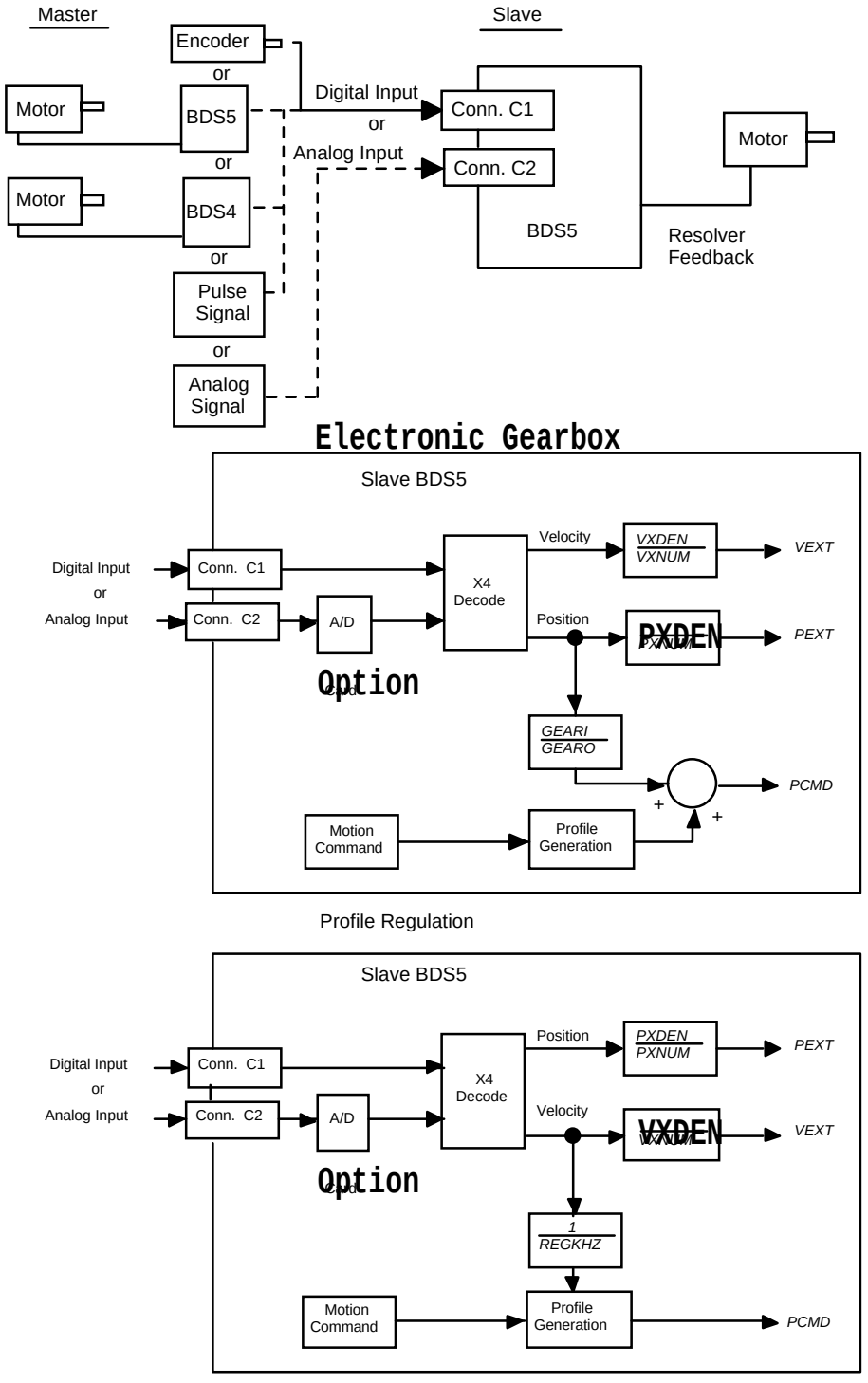


Figure 3.8. BDS5 Master/Slaving

```
GEAR  ON     ;ENABLE ELECTRONIC
             ;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.

3.8.13.5 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.

3.8.13.6 Gearbox, ACC/DEC, and Jogs

When the BDS5 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
```

3.8.14 Profile Regulation

This section describes profile regulation, one of the BDS5 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 BDS5, 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 BDS5 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.

3.8.14.1 REG & 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.



NOTE

The frequency of the external input should always be less than REGKHZ.

3.8.14.2 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 (that is, 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).

3.8.14.3 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 BDS5, as shown in the *Installation and Setup Manual*, "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.

3.8.15 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 BDS5 system. Refer to the *Installation and Setup Manual* and the model number to determine the resolution of your system. Select the encoder as follows:

Table 3.9. 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 BDS5 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.

3.8.16 CONTINUE

The CONTINUE command is provided as a controlled way to turn off master/slave position control. The CONTINUE command tells the BDS5 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
```

the 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 considerable. 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
```

the CONTINUE command would change the velocity command to the average velocity command over the previous 50 milliseconds. CONTINUE always sets SEG to 1.

The BDS5 provides several control loops. These loops, or control algorithms, allow you to select the best control method for your applications.

3.9 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.

BDS5 control loops have one or two tuning variables. All BDS5 loops follow the convention that larger constants provide higher gain. Each BDS5 loop is

described below and shown on the drawing at the end of this chapter.

3.9.1 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 hi
gh speed. If KF is large enough, then a high velocity command can be generated with little or no position error. The BDS5 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 BDS5 will force TQ off.



NOTE

When PL is turned on, TQ is turned off automatically.

3.9.2 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.

3.9.2.1 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 on 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.

3.9.2.2 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.

3.9.3 Torque Command

In a few applications, the BDS5 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 BDS5 is running in Torque command mode. The BDS5 will turn PL off when TQ is turned on.



NOTE

When TQ is turned on, PL is forced off

3.9.4 Power-Up Control Loops

The BDS5 has, at power-up, the following settings:

- Position loop enabled (PL on).
- No feed-forward (KF=0)
- Integrating Velocity Loop (PROP off, TQ off)

These settings meet the requirements of a large number of applications. Figure 3.9 shows each of the five BDS5 controller modes.

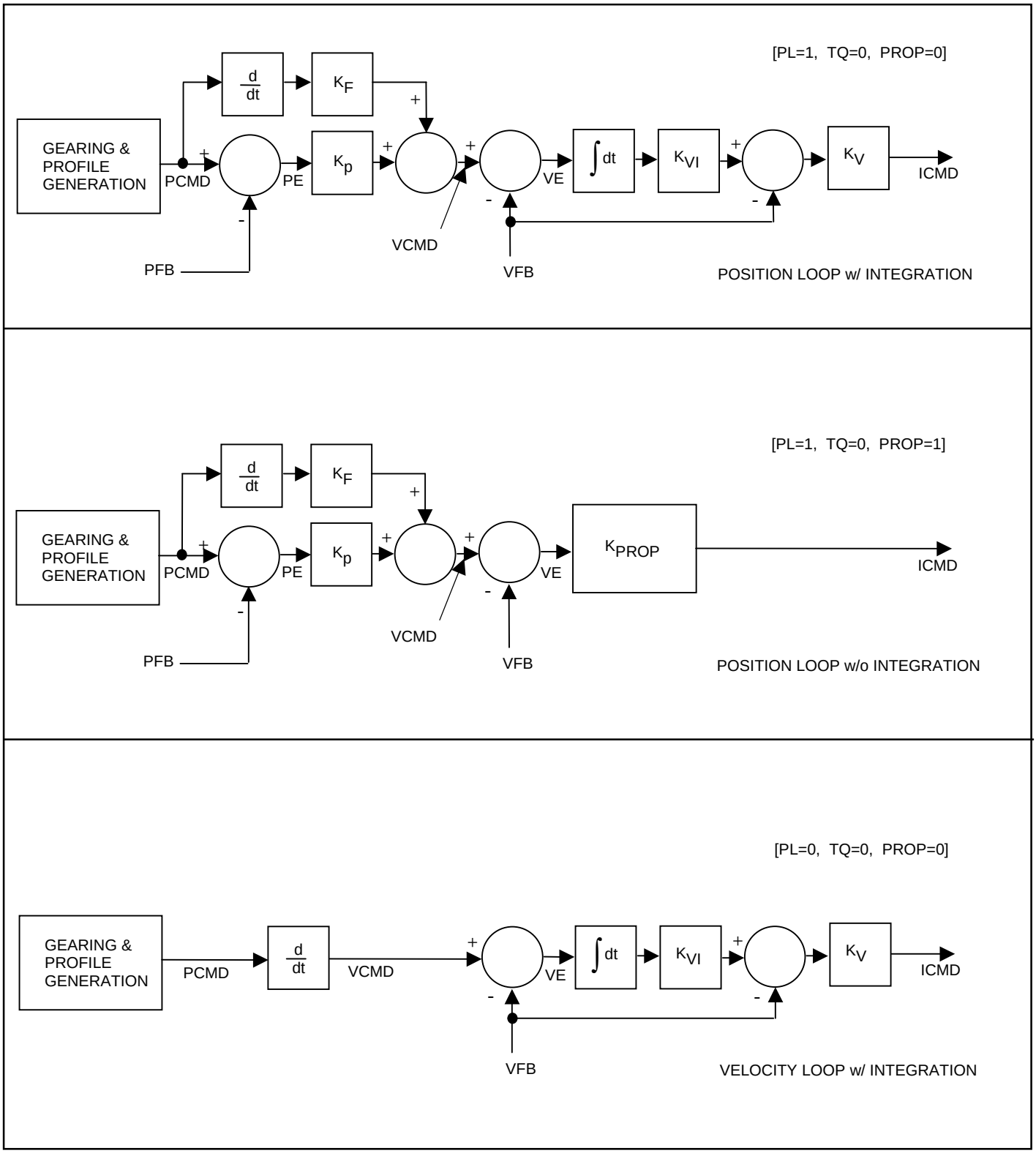


Figure 3.9. BDS5 Control Modes

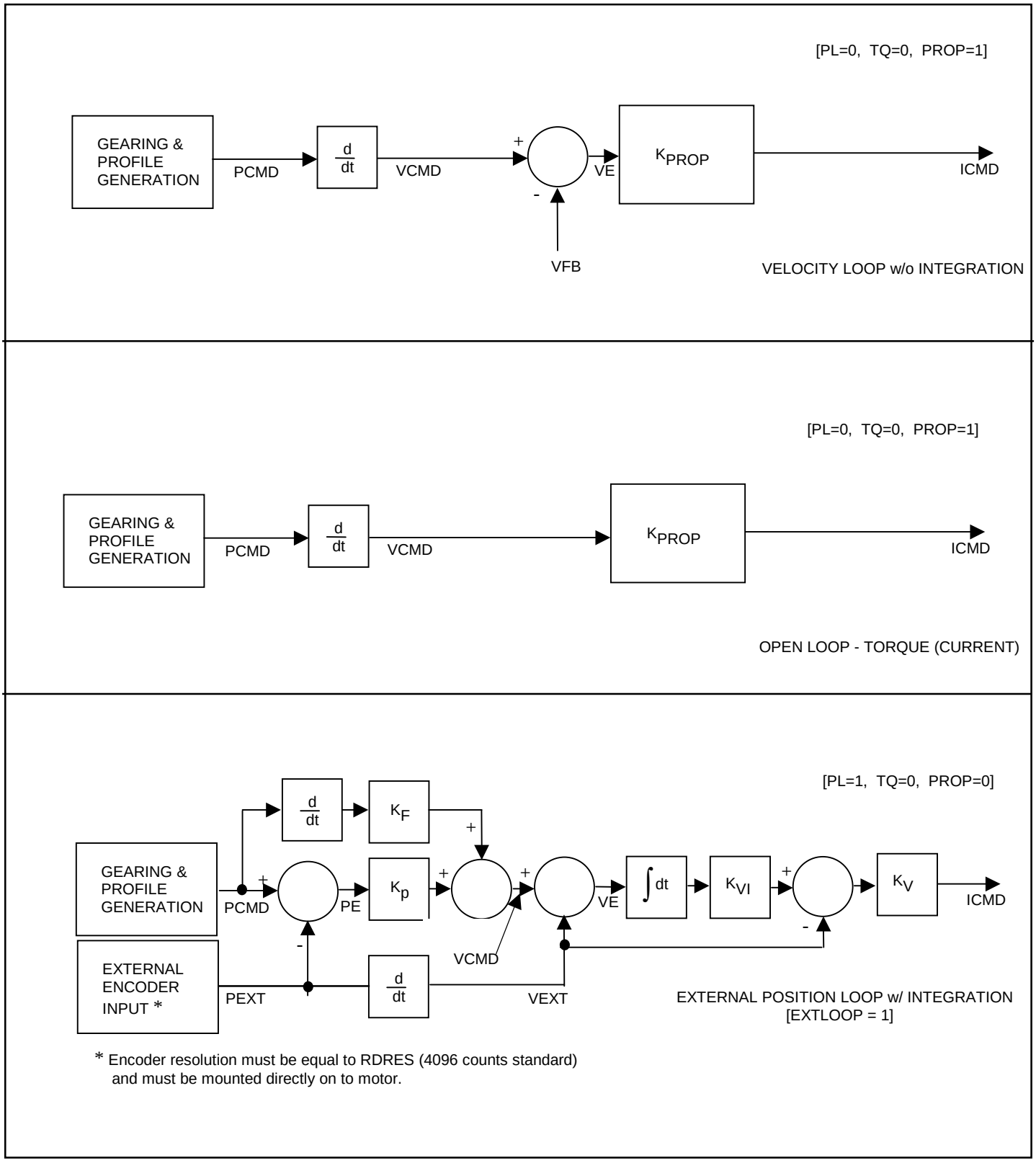


Figure 3.9. BDS5 Control Modes (Contd)

CHAPTER 4

USER PROGRAMS

4.1 INTRODUCTION

The information in this chapter will enable you to understand the capabilities of the system. You will also explore important consideraions that must be addressed before you implement your own application. Examples of programming techniques will aid you to develop your own applications.

4.2 PROGRAMMING TECHNIQUES

User programs are combinations of BDS5 commands which are stored in the BDS5 memory. These programs are stored in non-volatile RAM; they are not lost when the BDS5 is powered down. User programs are composed mostly 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 BDS5 Editor, which allows you to enter and modify programs from the terminal. If you have not already done so, read Chapter 3 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 BDS5 has a flexible language. You must follow proper programming principles to insure 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:

1. DO NOT PROGRAM SAFETY FUNCTIONS.



WARNING

Always hardwire personal safety functions. Never program these 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 BDS5 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.

2. USE CAUTION WHEN PROGRAMMING EQUIPMENT PROTECTION FUNCTIONS.



CAUTION

Use caution when programming equipment-protection functions. Programming errors can damage your equipment.

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 who are 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 (that is, 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.

4.2.1 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 BDS5 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."

Customer: "Okay. About how long do you think it will take?"

You: "I'll be in touch."

4.2.2 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.

4.2.3 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 BDS5 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. "Sample Flowchart" is a flowchart for this application.

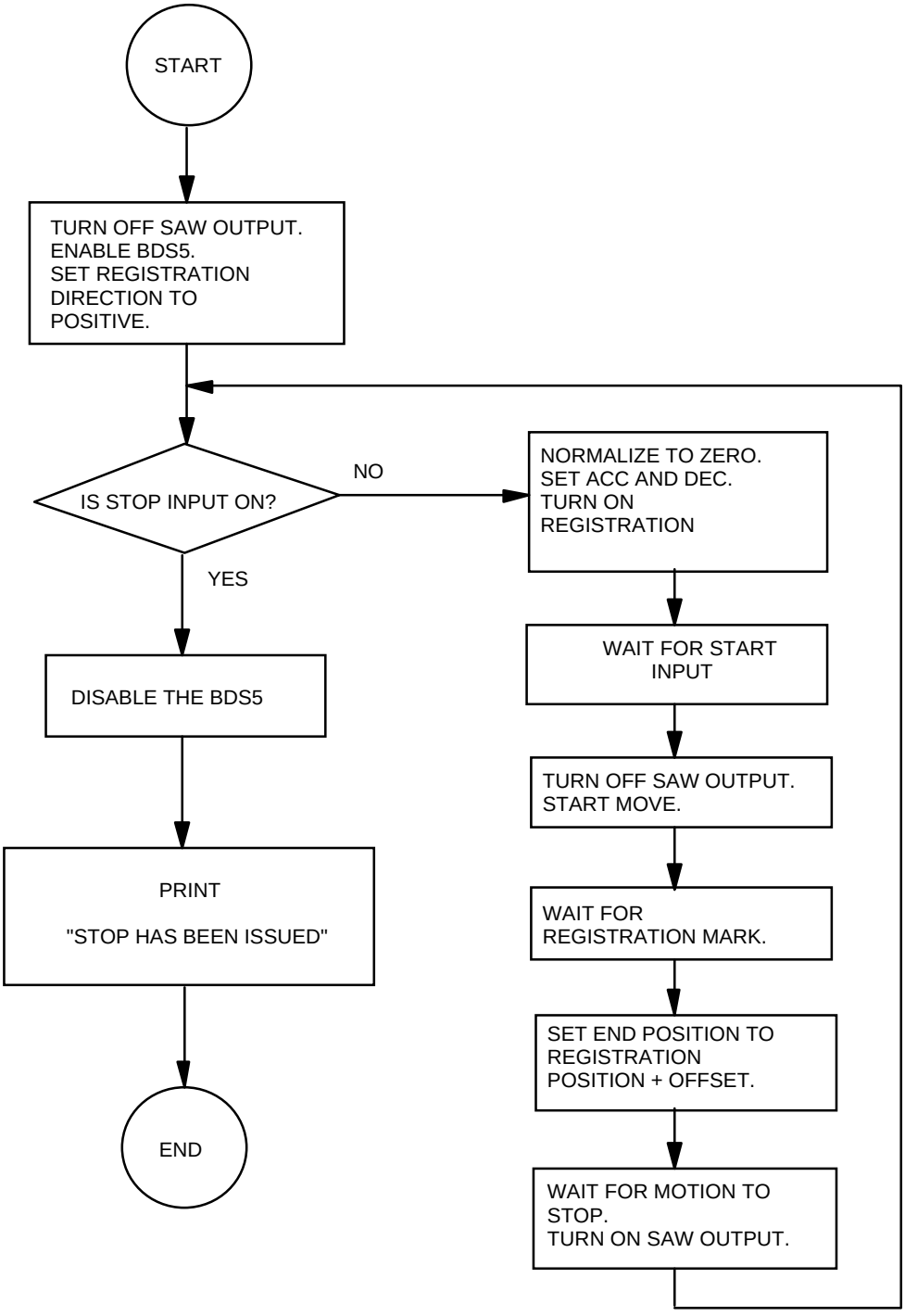


Figure 4.1 Sample Flowchart

4.2.4 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 BDS5

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 BDS5
P  "STOP HAS BEEN ISSUED"
B                       ;STOP EXECUTION
```

4.2.5 Customer Service

If you need help with software or understanding BDS5 functions, you can contact the Regional Industrial Drives' Sales Office. Ask for the Sales Applications Engineer. Please observe the following procedure:

1. Contact Industrial Drives for each new problem. Occasionally, an applications sales representative may refer you to the Engineering Department if necessary. However, if you call later with a new problem, please ask for a 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 application 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, please bear in mind that Industrial Drives is committed to helping you. BDS5 software support is provided by:

1. Helping you organize your program.

2. Explaining proper programming practices.

3. Discussing BDS5 functions.

Contact the local Industrial Drives sales application representative. All Regional Sales Offices are listed in Appendix I of this manual.

4.3 EDITING

Writing or modifying a program is called editing. There are two ways you can edit a BDS5 program.

The BDS5 has a simple editor which is built in, or resident. As an alternative, you can edit your program on a computer and transmit it to the BDS5. 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 BDS5 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 BDS5 be replaced or multiple BDS5's be programmed.

4.3.1 Motion Link Editor

Chapter 3 provided 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 communication with the BDS5 as discussed in Chapter 2.

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 BDS5. 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 3 for a complete description of Motion Link.

4.3.2 BDS5 Resident Editor

If you are not using the BDS5 Editor, skip ahead to the next section, "Building a Program." The BDS5 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 BDS5 Editor, type:

```
ED
```

When you are in the Editor, the BDS5 will respond with the editor prompt:

```
E->
```

To exit the Editor, press the escape key.

4.3.2.1 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 BDS5 Editor:

```
P  BEG  END
```

The BDS5 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
```

and the BDS5 will print and go to line 1. If you want to print the current line, then do not specify a line. For example:

```
P
```

prints the current line. 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 like:

```
BAD ENTRY
```

4.3.2.2 Next Line

If you enter an empty line, then the BDS5 goes down one line in the program and prints that line. The empty line is entered by pressing only the enter key. This makes it easy to move down through the program.

4.3.2.3 Password (PASS)

The BDS5 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
```

and the BDS5 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.

4.3.2.4 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. You exit the Insert mode by pressing the escape key or entering an empty line. For example, type in a line as follows. Type:

```
P  BEG        ;GO TO THE BEGINNING OF
THE PROGRAM
I             ;ENTER THE INSERT MODE
```

and the BDS5 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
```

and the BDS5 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.

4.3.2.5 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 BDS5 should respond with:

```
FIND WHAT?
F->
```

then type:

```
EDITOR
```

and 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 BDS5 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->
```

4.3.2.6 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
```

and the BDS5 should respond:

```
1       ;TEST LINE FOR LEARNING ABOUT
        ;THE EDITOR
```

Now use the C command to change "EDITOR" to "BDS5 EDITOR." Type:

```
C
```

and the BDS5 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 BDS5 will respond with:

```
CHANGE TO WHAT?
C->
```

Now type:

```
BDS5  EDITOR
```

The BDS5 will change the line to read:

```
1       ;TEST LINE FOR LEARNING ABOUT
        ;THE BDS5 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.

4.3.2.7 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.

4.3.2.8 Size

The BDS5 program memory has space for about 16000 characters. If you want to see how much memory is left, type:

```
SIZE
```

The BDS5 will respond with:

```
65 % LEFT
```

which means the available space is about 65%. If you try to enter a program larger than the BDS5 can store, an error will be generated.

4.3.2.9 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 BDS5 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 BDS5 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, which is discussed later in this Chapter, will also reset the program so that it is no longer corrupt, although it will not clear the password.

4.4 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.

4.4.1 Basic Commands

4.4.1.1 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.

4.4.1.2 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 BDS5 searches the entire program for some types of errors. If, after you enter a RUN command, an error is detected, the BDS5 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.

4.4.1.3 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.

4.4.1.4 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 short program below:

```
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
```

It is a good programming practice to avoid the use of GOTO commands in favor of Block-IF's, Quick-IF's, and GOSUB's. This practice makes programs more readable and easier to modify.

4.4.1.5 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
```

4.4.2 CONDITIONAL COMMANDS

The BDS5 provides several conditional commands which 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 BDS5 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 4.1. BDS5 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

4.4.2.1 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
```

4.4.2.2 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. BDS5 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.

4.4.2.3 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
```



NOTE

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
```

4.4.2.4 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, and thus 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... 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:

Table 4.2. Block-IF Restrictions and Options

Each IF/ELIF/ELSE/ENDIF set...
...must have one and only one IF.
...may have any number of ELIF's.
...need not have any ELIF's.
...may have one ELSE.
...need not have an ELSE.
...must have one and only one ENDIF.

4.4.2.5 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 4.3. 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.

4.4.2.6 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 BDS5 ignores the indentation.

4.4.2.7 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.



NOTE

You cannot GOTO the middle of an IF/ENDIF set. You should never execute a RET from between an IF and ENDIF.

4.5 USING THE GENERAL PURPOSE INPUTS

General purpose inputs can be used to control the program. From Chapter 3 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 On?	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 On?	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.

4.6 INTERFACING WITH THE OPERATOR

This section covers interfacing via the serial port (Connector C5). Often, it is necessary to have the BDS5 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 BDS5 serial I/O instructions.

4.6.1 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 BDS5 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.

4.6.1.1 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 BDS5 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 BDS5 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).

4.6.1.2 Printing Decimal Points

You can also use the BDS5 to print a decimal point. The BDS5 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 BDS5 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 BDS5 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 than all the digits which 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 digits to be printed:

```
P  "SPEED = "  VFB[5.3.1]  " KRPM"
```

This command would produce:

SPEED = 1.9 KRPM

.

So the general format 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.

4.6.1.3 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 BDS5 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."

4.6.1.4 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 BDS5 to print:

```
X2 =
00000000000000000000000001111111B
X2 = 001111111B
```

4.6.1.5 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 BDS5 variable. This print format allows you to print the previous answer on the screen the way it was entered.

4.6.1.6 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]
```

4.6.1.7 Printing ASCII Characters

The BDS5 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 BDS5 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 BDS5 removes the eighth bit before transmitting the character. For example:

```
P  65[C]  " IS THE SAME AS "  128+65[C]
```

since the BDS5 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 BDS5 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.

4.6.1.8 Printing Control Characters

The BDS5 uses the standard ASCII character set as shown in Appendix B. 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 BDS5 cannot print ASCII 0.

As Appendix B 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 B shows, <Control>G produces 07 or the ASCII bell. You can use the BDS5 to produce unprintable characters by preceding the appropriate character with the carat (^) to signify an unprintable character. For example, the following BDS5 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 BDS5 will print the carat and the 1 ("^1"). Only ^A to ^Z, ^[, ^/, ^], ^^ and ^_ are allowed.

4.6.1.9 Cursor Addressing

Many displays allow you to address the cursor. For example, the DEP-01 from Industrial Drives 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 BDS5 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 BDS5 serial port is connected to the DEP-01, "X" will be printed on space #0.

```
P "^[^A^HX MARKS THE FIRST SPACE"
```

4.6.1.10 Printing BDS5 Status (PS)

The PRINT STATUS (PS) command is like the P command except that it appends the BDS5 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:

Table 4.4. Printing BDS5 Status

Status	Explanation
OFF	BDS5 is OFF
READY	BDS5 is ready, but REMOTE is OFF.
ACTIVE	BDS5 is active, but no motion.
FAULT	BDS5 has a fault condition.
JOG	BDS5 is jogging.
PROFILE	BDS5 is executing profile.
GEAR	BDS5 is in gear mode.

You can use all formats and combinations with PS that you did with P. These results are identical except that the BDS5 status is appended onto the line.

4.6.2 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.

4.6.3 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 BDS5 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 BDS5 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
```

4.6.3.1 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 BDS5 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 BDS5 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 BDS5 uses the program limits. Appendix E lists all variables and their program limits.

4.6.3.2 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 BDS5. 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 BDS5; 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 BDS5 motion commands process it).

4.6.4 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. Otherwise, the serial port is not 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
```

4.7 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.

4.7.1 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 BDS5 to enter the following program:

```
29$
P  "TURN I1 ON"
H I1 ON
P  "I1 IS NOW ON"
B
```

Now 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.

4.7.2 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.

4.7.3 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.

4.8 MULTI-TASKING

Multi-tasking is an important feature of the BDS5. 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 BDS5 has 6 different task levels as shown on "Multi-Tasking Overview." 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.

4.8.1 Multitasking and Autobauding

If you set the BDS5 to autobaud, multi-tasking will not be enabled until communications have been established. This means that the BDS5 will not operate if a terminal or computer is not present. Therefore, you normally will want to disable autobauding by turning ABAUD off.



NOTE

Turn ABAUD off if you plan to use multi-tasking. The BDS5 will remember that ABAUD is off through power-up.

4.8.2 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.

4.8.3 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 BDS5 returns to the Interactive mode. However, if you end an alarm with the END command, the alarm stops, but the other tasks continue running.

4.8.4 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 BDS5 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 BDS5 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 multi-tasking without running a particular program, type:

```
RUN
```

without entering a label.

Table 4.5. Multi-Tasking Overveiw

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 BDS5 and Establixh Comminication	Initialize BDS5 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

The following two tables show how to turn multi-tasking on and off:

Table 4.6. How to Enable Multi-Tasking

1.	Run any label (Type "RUN <label>").
2.	Run multi-tasking (Type "RUN").
3.	Include a POWER-UP$ label and power-up.

Table 4.7. 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.

4.8.5 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.

4.8.5.1 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 BDS5 can store up to two MI or MA commands. This was called *buffering* in Chapter 3. 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. It 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 BDS5 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 BDS5 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.

4.8.5.2 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:

Table 4.8. Four Idling Commands

Wait (W)
Dwell (D)
Hold (H)
Input (INPUT)

For example, you can modify the above program to make one move, then run the background routine until motion has stopped. Use the BDS5 Editor to enter this 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 BDS5 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.

4.8.5.3 Avoiding Idling

You can avoid idling the BDS5 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.

4.8.6 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 BDS5 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 BDS5, 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 BDS5
OUT = 0             ;TURN OFF ALL
                    ;OUTPUTS
B                   ;BREAK EXECUTION
```

4.8.6.1 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.

4.8.6.2 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 BDS5. Otherwise, an INPUT command from another task may idle the alarm indefinitely.

4.8.7 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 DEP-01 Data Entry Panel from Industrial Drives 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 multi-tasking 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 BDS5 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.

4.8.7.1 Using Variable Input with Profiles

You can use the variable input routine while the BDS5 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 that are 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 block of lines.

4.8.7.2 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%.

4.8.8 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 from "Multi-Tasking Overview" 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$, and MANUAL$.

4.8.8.1 Power-Up Routine (POWER-UP$)

On power-up, the BDS5 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 BDS5 for a few seconds and power-up again. After establishing communications, the BDS5 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, then the BDS5 powers-up in the Interactive mode. If the BDS5 is set to autobaud, then 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.

4.8.8.2 Error Handler (ERROR$)

When a serious error occurs, the BDS5 breaks execution of your program and checks your program to see if you entered ERROR$. If you did, the error handler (that is, the routine that follows the ERROR$) is executed. All multi-tasking is suspended, including alarms, when the error handler is being executed.

4.8.8.3 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 BDS5 to begin executing one cycle of the auto program. The AUTO program begins at AUTO$. CYCLE READY is a hardware output that indicates the BDS5 is ready to run another cycle of the AUTO program.

The following conditions must be met for the BDS5 to execute the AUTO program. When these conditions are met, the CYCLE READY output (Connector C7, Pin 23) will turn on. Otherwise, it will be off.

Table 4.9. To Execute AUTO$...

1. Multi-tasking 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 BDS5 will begin executing the user program at AUTO$, and CYCLE READY output will turn off.

4.8.8.4 Manual Program (MANUAL $)

The following conditions must be met for the BDS5 to execute the MANUAL program. When these conditions are met, the BDS5 will begin executing label MANUAL$.

Table 4.10. To Execute MANUAL$...

1. Multi-tasking 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 BDS5 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.

4.8.8.5 Typical AUTO/MANUAL Programs

Drawing A-84983 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 BDS5 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.

4.8.9 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 multi-tasking 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.

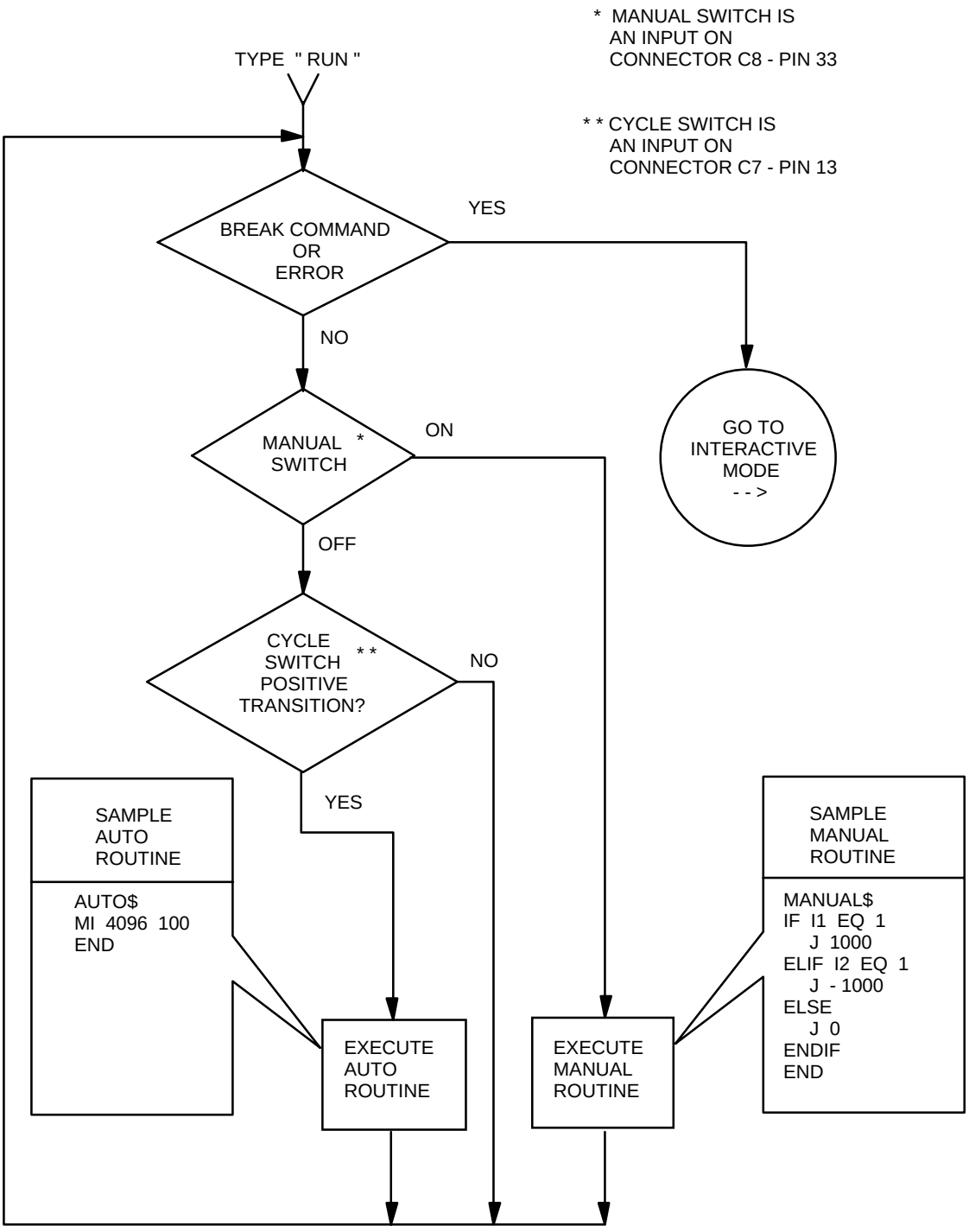


Figure 4.2. Auto/Manual Mode Flowchart

4.8.9.1 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%.

4.9 UNITS

The BDS5 provides user units so that both you and the machine operator can work in units that are convenient. The BDS5 allows you to define the units of acceleration, current, velocity, and position for your machine. Also, if your BDS5 has an external input, you can define the units of external position and external velocity.

4.9.1 User Units

The BDS5 uses internal units, called *BDS5-basic units*, that are very inconvenient to use. For example, velocity is in (1/65.536)*counts/second. User unit constants scale the BDS5-basic units. For example, if you type:

```
VOSPD = 1000
```

the 1000 is multiplied by VNUM/VDEN before it is stored in the BDS5 memory. Your BDS5 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 4.11. 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/(sec×sec)	ANUM=4474 ANUM=4272	ADEN=1000 ADEN=100

4.9.1.1 Current Units

The BDS5 commands current with a 12-bit digital-to-analog converter (DAC). The BDS5-basic current unit is 1/4095th of full-scale current. Full-scale current refers to the peak rating of your BDS5, not the continuous rating. For example, the peak rating of a 6 Amp BDS5 is 12 Amps.

The conversion constants that determine user current units are INUM, current units numerator, and IDEN, current units denominator:

$$\text{ILIM[basic units]} = \text{ILIM[user units]} \times \frac{\text{INUM}}{\text{IDEN}}$$

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 3, you typed:

```
ILIM=100                ;SET ILIM TO 100%
```

The BDS5 converted the 100% to 4095 BDS5-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 BDS5 converted the 4095 BDS5-basic units to 100% by multiplying by IDEN and dividing by INUM.

4.9.1.2 Other User Units

BDS5-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 4.12. 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 BDS5 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 BDS5-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 E, which lists all variables and the units associated with them.

4.9.1.3 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). Drawing A-84866 shows how external position and velocity come into a slave BDS5 and are displayed as PEXT and VEXT.

If the external input is a system with the same resolution as your BDS5, set external units as follows:

Table 4.13. Setting External Units in Master/Slave Systems

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.

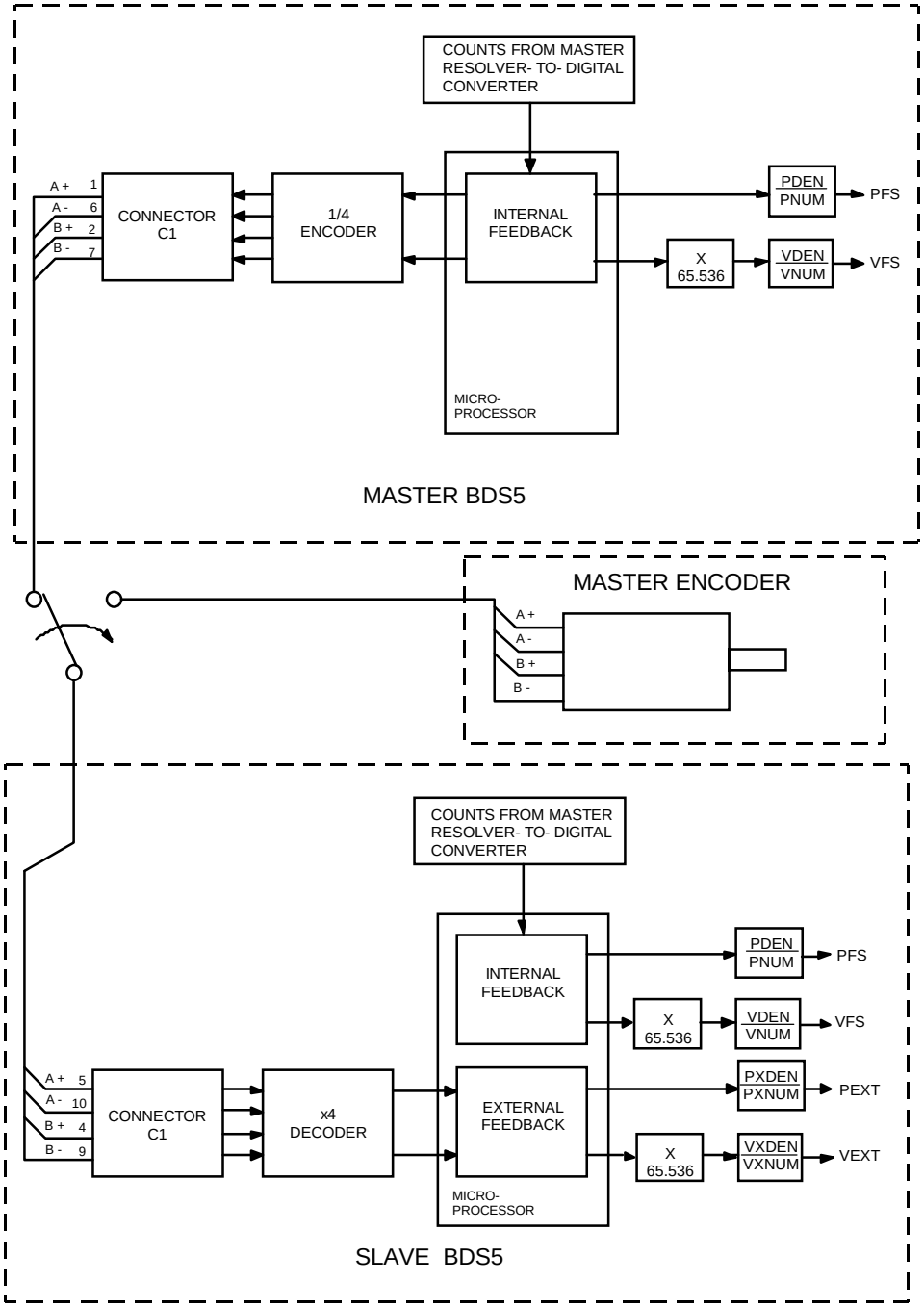


Figure 4.3. Master/Slave Block Diagram

4.9.2 Machine Specific Units

The BDS5 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 4.14 and 4.15 are for position, velocity, and acceleration units based.

Table 4.14. English Conversion (12-bit R/D Only)

POSITION UNITS $\frac{\text{PNUM}}{\text{PDEN}} = 4096 \times \left[\frac{\text{Motor Movement (In Revolutions)}}{\text{Machine Movement (In Your Units)}} \right]$
VELOCITY UNITS $\frac{\text{VNUM}}{\text{VDEN}} = 4473.92 \times \left[\frac{\text{Motor Velocity (In Rev / Min)}}{\text{Machine Velocity (In Your Units)}} \right]$
ACCELERATION UNITS $\frac{\text{ANUM}}{\text{ADEN}} = 4.47392 \times \left[\frac{\text{Motor Acceleration (In RPM / Sec)}}{\text{Machine Acceleration (In Your Units)}} \right]$

Table 4.15. Metric Conversion (12-bit R/D Only)

POSITION UNITS $\frac{\text{PNUM}}{\text{PDEN}} = 651.8971 \times \left[\frac{\text{Motor Movement (In Radians)}}{\text{Machine Movement (In Your Units)}} \right]$
VELOCITY UNITS $\frac{\text{VNUM}}{\text{VDEN}} = 712.047 \times \left[\frac{\text{Motor Velocity (In Rad / Sec)}}{\text{Machine Velocity (In Your Units)}} \right]$
ACCELERATION UNITS $\frac{\text{ANUM}}{\text{ADEN}} = 0.712047 \times \left[\frac{\text{Motor Acceleration (Rad / Sec / Sec)}}{\text{Machine Acceleration (Your Units)}} \right]$

The procedure to determine PNUM and PDEN is as follows:

A. Select Table 4.14 (revolutions) or 4.15 (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 4.14 (RPM) or 4.15 (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 4.14 (RPM/sec) or 4.15 (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.

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 4.16 has been provided to assist in calculating the conversion constants.

Table 4.16. External Units Conversion

EXTERNAL POSITION UNITS $\frac{\text{PXNUM}}{\text{PXDEN}} = \left[\frac{\text{External Input (In Counts)}}{\text{Machine Movement (In Your Units)}}\right]$
EXTERNAL VELOCITY UNITS $\frac{\text{VXNUM}}{\text{VXDEN}} = 65.535 \times \left[\frac{\text{External Input (In Counts / Sec)}}{\text{Machine Velocity (In Your Units)}}\right]$

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 4.16 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 4.16 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 4.14.

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 4.14 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 4.14.

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 4.14 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 4.14.

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 4.14 under ACCELERATION UNITS for the formula:

ANUM/ADEN = 4.47392 * (10 / 1) = 44.7392

Select ANUM and ADEN:

ANUM = 447392 ADEN = 10000

The BDS5 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.

4.9.3 Position Rotary Mode, ROTARY, & PROTARY

The BDS5 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 BDS5 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 $0 \leq PFB \leq PROTARY$ before turning ROTARY ON. Enable the Rotary mode by typing:

```
ROTARY ON
```

4.9.3.1 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 4.14,

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.

4.9.3.2 Rotary Mode and Absolute Moves

When the BDS5 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.

4.10 SERIAL COMMUNICATIONS

This section discusses details of BDS5 serial communications. This includes autobauding, multidrop connections, and transferring your program to and from the BDS5. If you are using Motion Link, the Industrial Drives software package for the BDS5, 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.

4.10.1 Autobauding

It is not necessary to set the baud rate on the BDS5 directly. Once the BDS5 is properly connected, it can determine the terminal's baud rate, then set its own baud rate accordingly. This is called autobauding. After the BDS5 determines the correct baud rate, it will store this rate away in BAUD. The BDS5 will flash the CPU light to indicate that it is autobauding. In order for the BDS5 to determine the baud rate setting on your terminal, you must press the enter key several times. Press only the enter key; otherwise the BDS5 will not autobaud correctly. The system will only autobaud during power-up.

4.10.1.1 Setting the BDS5 to Autobaud

There are three ways to set the BDS5 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 (say, 1000).

4.10.1.2 Autobauding and MOTION

If the MOTION input is off during power-up, the BDS5 will autobaud. (Note that this also sets ADDR to zero.) This allows you to command autobaud without being able to communicate with the BDS5. 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.

4.10.1.3 Enabling Autobaud with ABAUD

The autobaud software switch (ABAUD) is the usual way to tell the BDS5 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 BDS5 to autobaud when the unit is powered-up, then turn ABAUD off. This is important if you want the BDS5 to run the Power-Up Label (POWER-UP$), because if ABAUD is on, the BDS5 will not execute the program until communications have been established.

4.10.1.4 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 BDS5 will autobaud. After a successful autobaud, an error is generated indicating that the baud rate was out-of-range on power-up.

4.10.2 Prompts

The BDS5 issues a prompt when it is ready to receive a new command. Prompts are discussed in Chapter 3. The BDS5 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 BDS5 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.

4.10.3 Serial Watchdog

The BDS5 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 BDS5 and break the user program.

The serial watchdog is a safety feature that disables the BDS5 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 BDS5 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
```

4.10.4 Transmit/Receive Programs

The BDS5 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 the *Installation and Setup Manual* for communications format.

4.10.4.1 <BDS Command Receiving from the BDS5

The <BDS command is used to send the BDS5 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 BDS5 to store all your programs. Keep back-up copies elsewhere. The <BDS command will cause the BDS5 to transmit the entire user program to your computer. It cannot be issued in the Program mode. For example, from the terminal type:

```
<BDS
```

and the BDS5 will respond by printing out the entire user program.

4.10.4.2 The >BDS Command Transmitting to the BDS5

The >BDS command is used to send a new user program through the serial port to the BDS5. The transmission is ended by sending an escape character. Note that this command writes over the contents of

the user program stored in the BDS5. This command allows the program to be directly entered, presumably by a computer, to the BDS5. It cannot be issued in the program mode.



NOTE

The >BDS command writes over the entire user program.

The BDS5 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 the prompt before beginning to transmit a new line.

The >BDS command is password protected. If a password was set in the BDS5 Editor, then it must be given in the >BDS command.



NOTE

Typing in these examples will erase the user program in the BDS5. 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 BDS5.

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 BDS5 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 BDS5 issues a "USER PROGRAM CORRUPT" error. The >BDS command resets the user program memory, which eliminates this condition.

4.10.5 System Dump

The BDS5 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 BDS5 will provide pages of information including the program, all BDS5 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 BDS5. 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 ";" 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.

4.10.5.1 Version Dump

Your BDS5 will print out its firmware version at any time with the DUMP VERSION command:

```
DUMP VERSION
```

4.10.6 Multidrop Communications



NOTE

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 BDS5 is in Multidrop mode, each axis must have a unique address. This address is a prefix on all communications to and from the BDS5. 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 BDS5 powers-up in Multidrop mode it is "asleep." When asleep, the BDS5 continues to execute programs and control the motor properly, but it does not communicate over the serial line. The BDS5 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 BDS5 wakes up and communicates. The address is a backslash (\) followed by the ASCII character represented by ADDR. For example, if your BDS5 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
```



NOTE

This example sets the address to upper case A.

Setting ADDR to 65 makes this axis address "A" and automatically puts the BDS5 in Multidrop mode. This axis then waits for the "\A." After this, BDS5 is awakened and it remains awake until it receives a "\." 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 in the above example 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 allows you to know 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 4.17. BDS5 Prompts

Non-multidrop (ADDR=0)	Multidrop (ADDR = 65)
-->	A->
==>	A=>
s->	As>
t..	At.
e->	Ae>
i->	Ai>
f->	Af>
c->	Ac>

4.10.6.1 Broadcast

You may want to send all BDS5's on the serial line a command simultaneously. This is called a broadcast. You can broadcast by sending "*." In this case, all BDS5's execute the command. During a broadcast, none of the BDS5's can transmit, but all will receive and execute the command.

4.11 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 BDS5, 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 first you must 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 BDS5
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 5

DEBUGGING

5.1 INTRODUCTION

The information in this chapter will enable you to rectify problems you may have while programming the BDS5. When you write programs, you probably will 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 BDS5 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 BDS5 provides two execution modes to help you debug your program: Trace and Single-Step.

5.2 DEBUGGING MODES

5.2.1 Single-Step

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 BDS5 waits for you to press the ENTER key before executing the command. Use the nested-IF example given previously in this manual. 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 BDS5 variables from the Single-Step mode without stopping your program. For example, type:

```
P  X1
```

and the BDS5 should respond with:

```
        1
S-->
```

In this case, the BDS5 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.

5.2.2 Trace

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 BDS5 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.

5.2.2.1 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 BDS5 can transmit characters much faster than most PC's can process them. This can lead to delays of minutes between when the BDS5 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 BDS5 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.

5.3 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 4.8.5.2. 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 5.1. 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.*.

5.4 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 INDUSTRIAL DRIVES for help; 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.

5.5 SYNCHRONIZING YOUR PROGRAM

This section describes the functions and variables that allow you to synchronize the program to events, both external and internal.

5.5.1 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 BDS5 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
```

5.5.2 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 that when REG is on. RD delays are always regulated by the external frequency, even when REG is off.

5.5.3 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 BDS5 moves:

Table 5.2. 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 BDS5 is not commanding a profile. For example,

```
46$
MA 10000 1000
TIL SEG EQ 0 P "MOTION IN
PROGRESS"
B
```

continually prints a message until motion stops. Note that when SEG is zero, the BDS5 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
```

5.5.4 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 this program:

```
MI -50000 1000
TIL SEG EQ 4         ;BUG--DELAYS
                     ;INDEFINITELY
```

the TIL command would delay execution indefinitely because SEG would never equal 4. However,

```
MI -50000 1000
W 4                  ;BUG--DELAYS
                     ;UNTIL MOTION
                     ;STOPS
```

only delays until motion stops because the WAIT command does not delay program execution when

motion has stopped. 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. This is 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 last specified acceleration segment. For example, consider the Macro Move Example #1 in Chapter 3. 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
```

5.5.5 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 BDS5.

In the following example, two MI commands are entered and precalculated with GATEMODE on.

```
GATEMODE  ON       ;ENABLE GATING
MI  1000  100      ;PRECALC MOVES.
                   MOTION
MI  -1000          ;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.

5.6 HINTS

The following section lists some hints addressing the most common problems. Most result from a minor misuse or misunderstanding of a BDS5 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 BDS5.

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 BDS5 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 BDS5 RECORD function to record ICMD when a PE overflow occurs. If ICMD is saturating (that is, 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 BDS5 model numbers. If the voltage you apply to the BDS5 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 BDS5 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.

5.7 ERROR LOG

The BDS5 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.

5.7.1 Error Levels

The BDS5's response to an error depends on the error's severity. There are four levels of severity, listed below in increasing order:

Table 5.3. Error Severity Levels and Actions

1.	Errors which cause warnings.
2.	Errors which cause a program break and stop motion, in addition to Level 1 Actions.
3.	Errors which disable the system and set the FAULT LED, in addition to Level 2 Actions.
4.	Errors which disable almost all BDS5 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 BDS5 will respond with:

```
ERR  50 'J 100'              BDS5 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 BDS5 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.

5.7.2 DEP

If your BDS5 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 BDS5 provides the DEP switch, which, when turned on, cuts all error messages down to 40 characters. If your BDS5 prints to a DEP-01, type:

```
DEP  ON
```

5.7.3 Error History

The BDS5 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 BDS5 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.

5.7.4 Displaying Error Messages

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

```
ERR  50
```

The BDS5 responds with:

```
ERR  50 BDS5 INHIBITED
```

You may display messages for errors from 1 through 999. If you type in an error number that the BDS5 does not recognize, it will respond with:

```
ERROR NOT FOUND
```

A description of all errors is given in Appendix D.

5.7.5 Firmware Errors

Firmware errors are an indication of a serious problem with the BDS5. 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 D for information on these errors. Contact the factory should one of these errors occur.

CHAPTER 6

COMPENSATION

6.1 INTRODUCTION

The information in this chapter will enable you to compensate your motor for load conditions. Tuning is an important step in setting up and maintaining your BDS5 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.

6.2 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 BDS5 and a motor connected to a load) to a three-part criterion:

Table 6.1. Tuning Criterion

Noise Susceptibility
Response
Stability

Tuning is normally a laborious procedure requiring an experienced person. However, the BDS5 provides many tools to aid tuning, making it a much simpler process than it has been in the past.

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.

6.2.1 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 6.1, the graph shows the response of a BDS5 TACH signal (on Connector C2, Pin 2) to a square wave input when the system is critically damped.

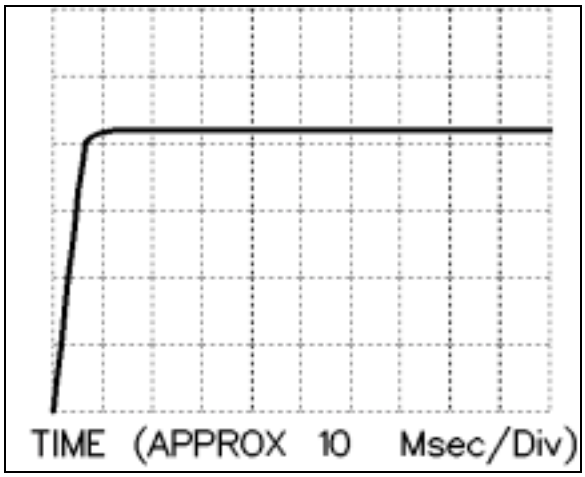


Figure 6.1. Critical Damping

6.2.2 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 6.2 shows a BDS5 slightly underdamped.

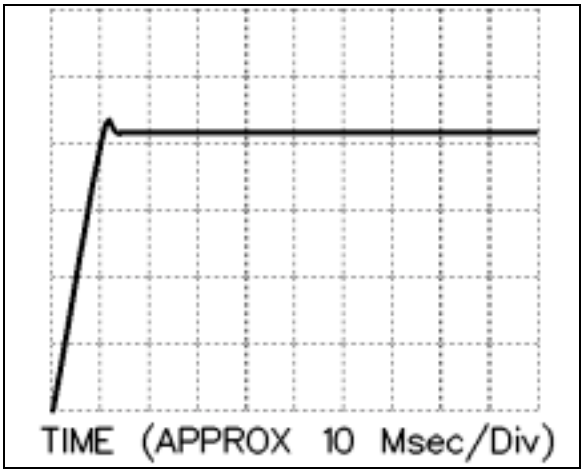


Figure 6.2. Underdamping

6.2.3 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 6.3 shows an overdamped system.

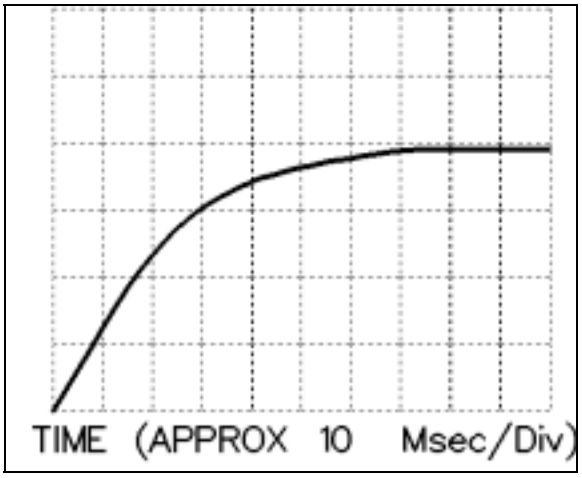


Figure 6.3. Overdamping

6.2.4 Ringing

When you are tuning the BDS5 you may tune it so that the response rings. Ringing is caused when you attempt to tune the BDS5 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 6.4, the graph shows a system that rings.

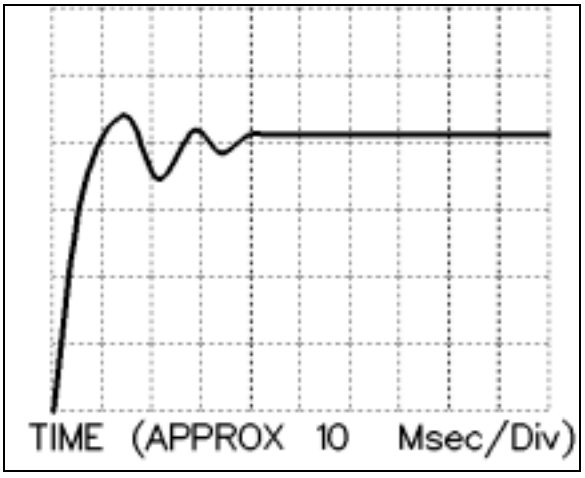


Figure 6.4. Ringing

6.3 TUNING



WARNING

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

The BDS5 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 have to adjust the tuning of your BDS5. The following section describes how you can re-tune your system.



NOTE

When tuning a system, it may be desirable to disable the BDS5 quickly. You can use K, the KILL command, to disable with a one-letter command.

The BDS5 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.



WARNING

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.

6.3.1 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 BDS5, 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
;BDS5 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 BDS5 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 BDS5 power-up.

6.3.2 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 BDS5 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.



CAUTION

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.



NOTE

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.

6.4 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:

Table 6.2. Allowed Tune Command Stability Settings

1	Slightly overdamped
2	Critically damped
3	Slightly underdamped



WARNING

The drive will be enabled and the motor will turn. Make sure the motor is secured. Even if the BDS5 is disabled, it will enable long enough to execute the TUNE command.

Enable the BDS5 and type this command:

```
TUNE 30 2
```

The BDS5 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 does not always provide an acceptable tuning. If not, you can tune the BDS5 yourself.

6.5 TUNING THE BDS5 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 BDS5 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.

6.5.1 Tuning the Velocity Loop



WARNING

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

Drawing A-84888 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 BDS5. Follow the instructions shown on Drawing A-84888. 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 on Drawing A-84888 in the the *Installation and Setup Manual*.

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 6.3.

Table 6.3. Velocity Loop Bandwith 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.

6.5.2 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:



WARNING

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 6.4 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 6.4. 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.

6.6 RECORD AND PLAY

The RECORD command allows you to record most BDS5 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

For example,

```
405$                ;BEGINNING LABEL
EN                  ;ENABLE BDS5
RECORD 500 1 VFB    ;RECORD VFB FOR
J  1000             ;1/2 SECOND JOG
B                   ;1000 RPM
```

Records the velocity response of the BDS5 to a JOG command.

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 BDS5 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.

6.7 PROBLEMS

Some times 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

6.7.1 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 BDS5 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 BDS5 with a higher current rating.
- Using a motor with more peak stall torque.

6.7.2 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.

6.7.3 Non-Linear Mechanics

BDS5 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 (that is, 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.

6.7.4 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.

6.7.5 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 BDS5. 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
```



NOTE

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

APPENDIX A

WARRANTY INFORMATION

Industrial Drives, a Kollmorgen Division, 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 Industrial Drives and its authorized dealers.

APPENDIX B

ASCII TABLE

The chart on the following pages is an ASCII Code and Hexadecimal conversion chart. The BDS5 doesnot 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 BDS5 does no support extended ASCII (128-255) Decimal to Hex Conversion.

APPENDIX C

SOFTWARE COMMANDS

C.1 EXPRESSIONS AND SYMBOLS

The following expressions and symbols are used in defining the syntax of the instruction set:

<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 operators, parentheses.

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 result 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 of 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.

C.2 COMMANDS

The following commands are the instructions used to program the BDS5.

;

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 BDS5 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$
A$  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 BDS5. This command turns off the variable READY. Refer to Drawing C-84732 for more information. Allowed from interactive mode, monitor mode, and user program.

Format: DIS

ED

Edit the user program. Allowed only from 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 BDS5 This command turns on the variable READY. Refer to Drawing C-84732 for more information. Allowed from 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 BDS5 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 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 BDS5 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 BDS5 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 5 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 5 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 BDS5 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), counter-clockwise (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 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  "BDS5"             ;PRINT "BDS5" 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 BDS5 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 Link's PLAYBACK,FROM BDS5 command rather than the BDS5 PLAY command. Motion Link formats, plots, and prints data in a much more readable form than does the BDS5.

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 the 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 BDS5 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 MULTITASKING

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.

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 task but lets other tasks proceed). Allowed only from the user program.

Format: W

Where Segment is a motion segment

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 BDS5 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 BDS5 program memory. This command destroys the old 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 from 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 D

ERROR CODES

D.1 INTRODUCTION

The BDS5's response to an error depends on the error's severity. There are four levels of severity, listed below in increasing order:

Table D.1. Error Severity Levels and Actions	
1.	Errors which cause warnings.
2.	Errors which cause a program break and stop motion, in addition to Level 1 Actions.
3.	Errors which cause the system to disable and set the FAULT Hardware Output, in addition to Level 2 Actions.
4.	Errors which disable almost all BDS5 functions (including communications) and flash the CPU LED to indicate the error number. These are called firmware errors.

See Chapter 5 for more information about error severity. The following is a complete list of errors generated by the BDS5.

D.2 HARDWARE FAULTS

D.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 BDS5 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 BDS5 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 BDS5 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 BDS5 will not communicate or run the user program. Check the +10 VDC input into the BDS5 (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 (PSR4/5 Connector C1, pins 2 and 3) is below 98 VAC (115 VAC less 15%).

D.2.2 BDS5 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 BDS5 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 BDS5.

ERROR 12 ***"OVER-CURRENT"*** ***SEVERITY 3***

The BDS5 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 BDS5.

ERROR 13 ***"OVER-SPEED"*** ***SEVERITY 3***

The BDS5 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 BDS5.

ERROR 14 ***"POWER BUS"*** ***SEVERITY 3***

The power supply high voltage bus has either an overvoltage fault or an undervoltage fault. This error breaks program execution and disables the BDS5.

ERROR 15 ***"COMP BOARD"*** ***SEVERITY 3***

You attempted to enable the BDS5 with the compensation board removed. Replace the compensation board. This error breaks program execution.

ERROR 17 ***"FEEDBACK LOSS"*** ***SEVERITY 3***

The BDS5 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 BDS5 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 BDS5 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.

ERROR 22 ***"+/- 12 VOLTS"*** ***SEVERITY 3***

The ±12 volts is out of tolerance. Contact the factory. This error breaks program execution.

D.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 BDS5 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 BDS5.

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 BDS5.

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.

ERROR 27 ***"R/D JUMPERS"*** ***SEVERITY 3***

Either the jumpers on your BDS5 MC2 card are incorrectly set or the wrong TL has been loaded. Contact the factory.

D.3 MOTION ERRORS

D.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 BDS5 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.

D.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 BDS5 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.

D.4 SOFTWARE ERRORS

D.4.1 Programming Modes or Motion Modes

ERROR 50 ***"DRIVE INHIBITED"*** ***SEVERITY 2***

You attempted to execute an instruction that required the BDS5 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 BDS5 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 BDS5, 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 BDS5 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 BDS5 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 BDS5 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 BDS5.

ERROR 65 ***"RECORD NOT READY"*** ***SEVERITY 3***

You entered a PLAY command when nothing had been recorded since the last time the BDS5 powered up.

D.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 BDS5 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.

D.4.3 Invalid Instructions or Entries

ERROR 79 ***"BAD FORMAT"*** ***SEVERITY 2***

You entered a format that the BDS5 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 in Chapter 4. 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 BDS5 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 B 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 C 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 C), 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 BDS5 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.

D.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 2^{31} or less than -2^{31}. This error breaks program execution.

ERROR 94 ***">2 PARENTHESES"*** ***SEVERITY 2***

The BDS5 evaluated an expression with more levels of parentheses than the BDS5 supports. Up to two levels of parentheses are allowed. This error breaks program execution.

ERROR 95 ***"UNEVEN PARENTHESES"*** ***SEVERITY 2***

The BDS5 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 2^{31} or less than -2^{31}. This error breaks program execution if the instruction is issued from the user program.

ERROR 97 ***"GEAR OVERFLOW"*** ***SEVERITY 3***

The BDS5 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 BDS5.

D.4.5 Communication Errors

ERROR 103 ***"BAUD RATE"*** ***SEVERITY 1***

The variable BAUD contains a value that is not supported by the BDS5. 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 BDS5.

D.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.

D.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 BDS5 has 4 levels of subroutines. This error breaks program execution.

ERROR 119 ***"RETURN w/o GOSUB"*** ***SEVERITY 2***

The BDS5 encountered a RET when it was not expecting one. This occurs when there are more returns than GOSUBs. This error breaks program execution.

D.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 BDS5 powers-up.

D.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 BDS5.

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 BDS5.

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 BDS5 is lost while editing the program. This error will break program execution. (See BDS5 Editor New Command to reset the user program; you will need to reload your program.)

ERROR 203 ***"AMPS BAD"*** ***SEVERITY 3***

The BDS5 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 BDS5.

ERROR 205 ***"MBUF OVRRUN"*** ***SEVERITY 3***

This is an internal error. Contact the factory. This error breaks program execution and disables the BDS5.

ERROR 206 ***"PROFILE OVERFLOW"*** ***SEVERITY 3***

This is an internal error. Contact the factory. This error breaks program execution and disables the BDS5.

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 BDS5.

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 BDS5.

ERROR 211-219 ***"INTERNAL 1-9"*** ***SEVERITY 3***

These are internal errors. Contact the factory. These errors break program execution and disable the BDS5.

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 BDS5.

APPENDIX E

VARIABLE QUICK REFERENCE

E.1 INTRODUCTION

This appendix lists all the variables on the BDS5. 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 BDS5 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 BDS5 for the particular motor it will be controlling. The MOTOR command changes these variables as necessary for the motor.

E.2 STANDARD VARIABLES

Table E.1. Standard Variables

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS[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 at end of selection 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	
LPF	Enable Low Pass Filter	Always	None	0,1
LPFHZ	Low Pass Filter Freq	Always	Hz	0-500

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
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
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	
PROP	Enable Prop. Mode	Always	None	0,1
PTRIP1	Position Trip Point #1	Always	POS	Long

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
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
VDEN	VEL Units Denominator	Always	None	Long
VE	Velocity Error	Never	VEL	

VARIABLE	DESCRIPTION	PROGRAM CONDITION	UNITS	PROGRAM LIMITS
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 E.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

E.3 INTERNAL VARIABLES

The following variables are internal variables and are not normally used by customers. They are set at the factory and program the BDS5 for the particular motor it will be controlling. The Motor command changes these variables as necessary for the motor.

Table E.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 F

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 BDS5 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 BDS5 is enabled.

2. PLIM, PL, and LPF are on.

3. TQ, and PROP are off.

4. No profiles are being calculated. That is, the BDS5 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 BDS5 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.

```
GOSUB  10              ;1.6 MSEC
GOTO  10               ;1.6 MSEC

JT  50000  1000        ;5.8 MSEC (CALCULATION TIME ONLY)
JF  50000  1000        ;5.8 MSEC (CALCULATION TIME ONLY)

MA  4096  100          ;5.5 MSEC (CALCULATION TIME ONLY)
MA  4096               ;5.0 MSEC (CALCULATION TIME ONLY)

MI  4096  100          ;5.8 MSEC (CALCULATION TIME ONLY)
MI  4096               ;5.0 MSEC (CALCULATION TIME ONLY)

MCI  10000  1000  200
MCI  1000  0           ;11.0 MSEC FOR ALL 3 COMMANDS
MCGO                   ; (CALCULATION TIME ONLY)

MRD 1000 100 CW        ;3.5 MSEC

O1  ON                 ;1.9 MSEC
O1  OFF                ;1.9 MSEC
OUT=OUT!0C8H           ;2.8 MSEC

NORM  0                ;2.0 MSEC

P  X1                  ;3.5 MSEC
P  X1[8]               ;3.5 MSEC
P  "X1="  X1           ;3.8 MSEC

RET                    ;0.8 MSEC

TIL  1  EQ  0          ;2.6 MSEC

X1=X2                  ;1.7 MSEC
X1=X2+1                ;2.2 MSEC
X1=X2-1                ;2.2 MSEC
X1=X2*100              ;2.3 MSEC
X1=X2/100              ;2.3 MSEC

ZPE                    ;1.0 MSEC

?  1  EQ  1  O1  ON    ;4.0 MSEC

IF  1  EQ  0           ;9.0 MSEC (ALL 7 LINES)
  X1=1
ELIF  1  EQ  0
  X1=2
ELSE
  X1=3
ENDIF

10$                    ;1.0 MSEC
```

GLOSSARY

Absolute Position

Position referenced to a fixed zero position.

Absolute Positioning

Refers to a motion control system employing position feedback devices (absolute encoders) to maintain a given mechanical location.

Absolute Programming

A positioning coordinate reference wherein all positions are specified relative to some reference, or "home" position. This is different from incremental programming, where distances are specified relative to the current position.

AC Adjustable-Speed Drive

All equipment required to adjust the speed or torque of AC electric motor(s) by controlling both frequency and voltage applied to the motor(s).

AC Servo Drive

A servo drive used to control either or both synchronous or induction AC motors.

Acceleration

The change in velocity as a function of time. Acceleration usually refers to increasing velocity and deceleration describes decreasing velocity.

Accuracy

A measure of the difference between expected position and actual position of a motor or mechanical system. Motor accuracy is usually specified as an angle representing the maximum deviation from expected position.

Actuator

A device which creates mechanical motion by converting various forms of energy to mechanical energy.

Adaptive Control

A technique to allow the control to automatically compensate for changes in system parameters such as load variations.

Ambient Temperature

The temperature of the cooling medium, usually air, immediately surrounding the motor or another device.

Amplifier

Electronics which convert low level command signals to high power voltages and currents to operate a servomotor.

ASCII

(American Standard Code for Information Interchange) This code assigns a number to each numeral letter of the alphabet. In this manner, information can be transmitted between machines as a series of binary numbers.

Back EMF

The voltage generated when a permanent magnet motor is rotated. This voltage is proportional to motor speed and is present regardless of whether the motor winding(s) are energized or un-energized.

Bandwidth

The frequency range in which the magnitude of the system gain expressed in dB is greater than -3 dB.

Baud Rate

The number of binary bits transmitted per second on a serial communications link (such as RS-232).

Bit (Binary Digit)

A unit of information equal to 1 binary decision or having only a value 0 or 1.

Block Diagram

A simplified schematic representing components and signal flow through a system.

Bode Plot

A plot of the magnitude of system gain in dB and the phase of system gain in degrees versus the sinusoidal input signal frequency in logarithmic scale.

Brownout

Low-line voltage at which the device no longer functions properly.

Brush

Conducting material which passes current from the DC motor terminals to the rotating commutator.

Brushless Servo Drive

A servo drive used to control a permanent magnet synchronous AC motor. May also be referred to as an AC Servo Drive.

Bus

A group of parallel connections carrying pre-assigned digital signals. Buses usually consist of address and data information and miscellaneous control signals for the interconnection of microprocessors, memories, and other computing elements.

Byte

A group of 8 bits treated as a whole with 256 possible combinations of ones and zeros, each combination representing a unique piece of information.

CAM Profile

A technique used to perform nonlinear motion electronically similar to that achieved with mechanical cams.

Characteristic Equation

1+GH = 0, where G is the transfer function of the forward signal path and H is the transfer function of the feedback signal path.

Circular Coordinated Move

A coordinated move where the path between endpoints is the arc of a circle.

Class B Insulation

A NEMA insulation specification. Class B insulation is rated to an operating temperature of 130 degrees centigrade.

Class H Insulation

A NEMA insulation specification. Class H insulation is rated to an operating temperature of 180 degrees centigrade.

Closed Loop

A broadly applied term relating to any system where the output is measured and compared to the input. The output is then adjusted to reach the desired condition. In motion control, the term is used to describe a system wherein a velocity or position (or both) transducer is used to generate correction signals by comparison to desired parameters.

Cogging

A term used to describe non-uniform angular velocity. Cogging appears as a jerkiness especially at low speeds.

Command Position

The desired angular or linear position of an actuator.

Commutation

A term which refers to the action of steering currents or voltage to the proper motor phases so as to produce optimum motor torque. In brush type motors, commutation is done electromechanically via the brushes and commutator. In brushless motors, commutation is done by the switching electronics using rotor position information typically obtained by hall sensors, a tachsyn, a resolver or an encoder.

Commutator

A mechanical cylinder consisting of alternating segments of conductive and insulating material. This cylinder used in DC motors passes currents from the brushes into the rotor windings and performs motor commutation as the motor rotates.

Compensation

The corrective or control action in a feedback loop system which is used to improve system performance characteristics such as accuracy and response time.

Compensation, Feedforward

A control action which depends on the command only and not the error to improve system response time.

Compensation, Integral

A control action which is proportional to the integral or accumulative time error value product of the feedback loop error signal. It is usually used to reduce static error.

Compensation, Lag

A control action which causes the lag at low frequencies and tends to increase the delay between the input and output of a system while decreasing static error.

Compensation, Lead

A control action which causes the phase to lead at high frequencies and tends to decrease the delay between the input and output of a system.

Compensation, Lead Lag

A control action which combines the characteristics of lead and lag compensations.

Compensation, Proportional

A control action which is directly proportional to the error signal of a feedback loop. It is used to improve system accuracy and response time.

Compliance

The amount of displacement per unit of applied force.

Computer Numerical Control

A computer-based motion control device programmable in a numerical word address format. A computer numerical control (CNC) product typically includes a CPU section, operator interface devices, input/output signal and data devices, software and related peripheral apparatus.

Control Systems or Automatic Control Systems

An engineering or scientific field that deals with controlling or determining the performance of dynamic systems such as servo systems.

Coordinated Motion

Multi-axis motion where the position of each axis is dependent on the other axis such that the path and velocity of a move can be accurately controlled. (Requires coordination between axes.)

Coupling Ratio

The ratio of motor velocity to load velocity for a load coupled to motor through a gear or similar mechanical device.

Critical Damping

A system is critically damped when the response to a step change in desired velocity or position is achieved in the minimum possible time with little or no overshoot.

Daisy Chain

A term used to describe the linking of several RS232C devices in sequence such that a single data stream flows through one device and on to the next. Daisy-chained devices usually are distinguished by device addresses which serve to indicate the desired destination for data in the stream.

Damping

An indication of the rate of decay of a signal to its steady state value. Related to setting time.

Damping Ratio

Ratio of actual damping to critical damping. Less than one is an underdamped system and greater than one is an overdamped system.

DC Adjustable-Speed Drive

All equipment required to adjust the speed or torque of DC motor(s) by controlling the voltages applied to the armature and/or field of the motors.

DC Drive

An electronic control unit for running DC motors. The DC drive converts AC line current to a variable DC current to control a DC motor. The DC drive has a signal input that controls the torque and speed of the motor.

Dead Band

A range of input signals for which there is no system response.

Decibel (dB)

A logarithmic measurement of gain. If G is a systems gain (ratio of output to input) then 20 log G = gain in decibels (dB).

Demag Current

The current level at which the motor magnets will be demagnetized. This is an irreversible effect which will alter the motor characteristics and degrade performance.

Detent Torque

The maximum torque that can be applied to an un-energized stepping motor without causing continuous rotating motion.

Dielectric Test

A high voltage breakdown test of insulation's ability to withstand an AC voltage. Test criterion limits the leakage current to a specified magnitude and frequency, applied between the specified test points.

Differential

An electrical input or output signal which uses two lines of opposite polarity referenced to the local signal ground.

Distributed Processing

A technique to gain increased performance and modularity in control systems utilizing multiple computers or processors.

DNC, Direct Numerical Control

Technique of transferring part program data to a numerical control system via direct electrical connection in place of paper tapes.

Drive

This is the electronics portion of the system that controls power to the motor.

Drive, Analog

Usually referring to any type of motor drive in which the input is an analog signal.

Drive, Digital

Usually referring to any type of motor drive in which the tuning or compensation is done digitally. Input may be an analog or digital signal.

Drive, Linear

A motor drive in which the output is directly proportional to either a voltage or current input. Normally both inputs and outputs are analog signals. This is a relatively inefficient drive type.

Drive, PWM

A motor drive utilizing Pulse-Width Modulation techniques to control power to the motor. Typically a high efficiency drive that can be used for high response application.

Drive, SCR

A DC motor drive which utilizes internal silicon controlled rectifiers as the power control elements. Usually used for low bandwidths, high power applications.

Drive, Servo

A motor drive which utilizes internal feedback loops for accurate control of motor current and/or velocity.

Drive, Stepper

Electronics which convert step and direction inputs to high power currents and voltages to drive a stepping motor. The stepping motor driver is analogous to the servo motor amplifier.

Duty Cycle

For a repetitive cycle, the ratio of an on time to total cycle time.

$$\text{Duty Cycle} = \frac{(\text{On Time})}{(\text{On Time} + \text{Off Time})} \times 100\%$$

Dynamic Braking

A passive technique for stopping a permanent magnet brush or brushless motor. The motor windings are shorted together through a resistor which results in motor braking with an exponential decrease in speed.

Efficiency

The ratio of power output to power input.

Electrical Time Constant

The ratio of armature inductance to armature resistance.

Electronic Gearing

A technique used to electrically simulate mechanical gearing. Causes one closed loop axis to be slaved to another open or closed loop axis with a variable ratio.

EMI: Electro-Magnetic Interference

EMI is noise which, when coupled into sensitive electronic circuits, may cause problems.

Encoder

A type of feedback device which converts mechanical motion into electrical signals to indicate actuator position. Typical encoders are designed with a printed disc and a light source. As the disc turns with the actuator shaft, the light source shines through the printed pattern onto a sensor. The light transmission is interrupted by the patterns on the disc. These interruptions are sensed and converted to electrical pulses. By counting these pulses, actuator shaft position is determined.

Encoder, Absolute

A digital position transducer in which the output is representative of the absolute position of the input shaft within one (or more) revolutions. Output is usually a parallel digital word.

Encoder, Incremental

A position encoding device in which the output represents incremental changes in position.

Encoder, Linear

A digital position transducer which directly measures linear position.

Encoder Marker

A ounce-per-revolution signal provided by some incremental encoders to specify a reference point within that revolution. Also known as Zero Reference signal or index pulse.

Encoder Resolution

A measure of the smallest positional change which can be detected by the encoder.

Explosion-proof

A motor classification that indicates a motor is capable of withstanding internal explosions without bursting or allowing ignition to reach beyond the confines of the motor frame.

Fall Time

The time for the amplitude of system response to decay to 37% of its steady-state value after the removal of a steady-state step input signal.

Feed Forward

A technique used to pre-compensate control a loop for known errors due to motor, drive, or lead characteristics. Provides improved response.

Feedback

A signal which is transferred from the output back to the input for use in a closed loop system.

Field Weakening

A method of increasing the speed of a wound field DC motor; reducing stator magnetic field instantly by reducing magnet winding current.

Filter (Control Systems)

A transfer function used to modify the frequency or time response of a control system.

Flutter

Flutter is an error of the basic cycle of an encoder per one revolution.

Following Error

The positional error during motion resulting from use of a position control loop with proportional gain only.

Form Factor

The ratio of RMS current to average current. This number is a measure of the current ripple in a PWM or other switch mode type of controller. Since motor heating is a function of RMS current while motor torque is a function of average current, a form factor greater than 1.00 means some fraction of motor current is producing heat but not torque.

Four Quadrant

Refers to a motion system which can operate in all four quadrants i.e. velocity in either direction and torque in either direction. This means that the motor can accelerate, run, and decelerate in either direction.

Friction

A resistance to motion caused by surfaces rubbing together. Friction can be constant with varying speed (coulomb friction) or proportional to speed (viscous friction) or present at rest (static friction).

Full Load Current

The armature current of a motor operated at its full load torque and speed with rated voltage applied.

Full Load Speed

The speed of a motor operated with rated voltage and full load torque.

Gain

The ratio of system output signal to system input signal.

Hall Sensors

A feedback device which is used in a brushless servo system to provide information for the amplifier to electronically commutate the motor. The device uses a magnetized wheel and hall-effect sensors to generate the commutation signals.

Holding Torque

Sometimes called torque, it specifies the maximum external force or torque that can be applied to a stopped, energized motor without causing the rotor to rotate continuously.

Home Position

A reference position for all absolute positioning movements. Usually defined by a home limit switch and/or encoder marker. Normally set at power up and retained for as long as the control system is operational.

Host Computer

An auxiliary computer system which is connected to a controller or controllers. The host computer in distributed control systems is frequently involved with controlling many remote and distributed motion control devices. It may also be used for off-line tasks such as program preparation, storage, and supervisory control and evaluation.

HP: Horsepower

One horsepower is equal to 746 watts. Since Power = Torque x Speed, horsepower is a measure of a motor's torque and speed capability (e.g. a 1 HP motor will produce 35 lb.-in. at 1800 rpm).

Hunting

The oscillation of the system response about a theoretical steady-state value.

Hybrid Stepping Motor

A motor designed to move in discrete increments or steps. The motor has a permanent magnet rotor and wound stator. These motors are brushless and phase currents are commutated as a function of time to produce motion.

Hysteresis

The difference in response of a system to an increasing or a decreasing input signal.

I/O: Input/Output

The reception and transmission of information between control devices. In modern control systems, I/O has two distinct forms: switches, relays, etc., which are in either an on or off state, or analog signals that are continuous in nature such as speed, temperature, flow, etc.

Idle Current Reduction

A stepping motor driver feature that reduces the phase current to the motor when no motor motion (idle) is commanded for a specified period of time. This reduces motor heating and allows high machine throughput to be obtained from a given motor.

Incremental Motion

A motion control term that is used to describe a device that produces one step of motion for each step command (usually a pulse) received.

Indexer

Electronics which convert high level motion commands from a host computer, programmable controller, or operator panel into step direction pulse streams for use by the stepping motor driver.

Inertia

The property of an object to resist changes in velocity unless acted upon by an outside force. Higher inertia objects require larger torques to accelerate and decelerate. Inertia is dependent upon the mass and shape of the object.

Inertial Match

An inertial match between motor and load is obtained by selecting the coupling ratio such that the load moment of inertia referred to the motor shaft is equal to the motor moment of inertia.

Inrush Current

The current surge generated when a piece of equipment such as a servoamplifier is connected to an AC line. This surge is typically due to the impulse charging of a large capacitor located in the equipment.

Instability

Undesirable motion of an actuator that is different from the command motion. Instability can take the form of irregular speed or hunting of the final rest position.

Lead Ball Screw

A lead screw which has its threads formed as a ball bearing race; the carriage contains a circulating supply of balls for increased efficiency.

Lead Screw

A device for translating rotary motion into linear motion, consisting of an externally threaded screw and an internally threaded carriage (nut).

Least Significant Bit

The bit in a binary number that is the least important, or having the least weight.

Limits

Properly designed motion control systems have sensors called limits that alert the control electronics that the physical end of travel is being approached and that motion should stop.

Linear Coordinated Move

A coordinated move where the path between endpoints is a line.

Linearity

For a speed control system it is the maximum deviation between actual and set speed expressed as a percentage of set speed.

Logic Ground

An electrical potential to which all control signals in a particular system are referenced.

Loop, Feedback Control

A control method that compares the input from a measurement device, such as an encoder or tachometer, to a desired parameter, such as a position or velocity and causes action to correct any detected error. Several types of loops can be used in combination (i.e. velocity and position together) for high performance requirements.

Loop Gain, Open

The product of the forward path and feedback path gains.

Loop, PID: Proportional, Integral, and Derivative Loop

Specialized very high performance control loop which gives superior response.

Loop, Position

A feedback control loop in which the controlled parameter is motor position.

Loop, Velocity

A feedback control loop in which the controlled parameter is mechanical velocity.

Master Slave Motion Control

A type of coordinated motion control where the master axis position is used to generate one or more slave axis position commands.

Mechanical Time Constant

The time for an unloaded motor to reach 63.2% of its final velocity after the application of a DC armature voltage.

Microstepping

An electronic control technique that proportions the current in a step motor's windings to provide additional intermediate positions between poles. Produces smooth rotation over a wide speed range and high positional resolution.

Mid-Range Instability

A phenomenon in which a stepping motor can fall out of synchronism due to loss of torque at mid-range speeds. The loss of torque is due to interaction between the motor's electrical characteristics and the driver electronics. Some drivers have circuitry to eliminate or reduce this phenomenon.

Most Significant Bit

The bit in a binary number that is the most important or that has the most weight.

Motor, AC

A device that converts electrical alternating current into mechanical energy. Requires no commutation devices such as brushes. Normally operated off commercial AC power. Can be single- or multiple-phase.

Motor, AC Asynchronous or Induction

An AC motor in which speed is proportional to the frequency of the applied AC. Requires no magnets or field coil. Usually used for non-precise constant speed applications.

Motor, AC Synchronous

Another term for brushless DC motor.

Motor Constant

The ratio of the motor torque to motor input power.

Motor, DC

A device that converts electrical direct current into mechanical energy. It requires a commutating device, either brushes or electronic. Usually requires a source of DC power.

Motor, DC Brushless

A type of direct current motor that utilizes electronic commutation rather than brushes to transfer current.

Motor, DC Permanent Magnet

A motor utilizing permanent magnets to produce a magnetic field. Has linear torque speed characteristics.

Motor, DC Wound Field

A direct current utilizing a coil to produce a magnetic field. Usually used in high power applications where constant horsepower operation is desired.

Motor, Stepping

A specialized AC motor that allows discrete positioning without feedback. Normally used for non-critical, low power applications, since positional information is easily lost if acceleration or velocity limits are exceeded. Load variations can also cause loss of position. If encoders are used, these limitations can be overcome.

NC, Numerical Control

Usually refers to any type of automated equipment or process used for contouring or positioning.

Negative Feedback

The type of feedbacks used in a closed loop system where the output value is inverted and combined with the input to be used to stabilize or improve system characteristics.

No Load Speed

Motor speed with no external load.

Open Collector

A term used to describe a signal output that is performed with a transistor. An open collector output acts like a switch closure with one end of the switch at ground potential and the other end of the switch accessible.

Open-Loop System

A system where the command signal results in actuator movement but, because the movement is not sensed, there is no way to correct for error. Open loop means no feedback.

Operator Interface

A device that allows the operator to communicate with a machine. This device typically has a keyboard or thumbwheel to enter instructions into the machine. It also has a display device that allows the machine to display messages.

Optically Isolated

A system or circuit that transmits signals with no direct electrical connection. Used to protectively isolate electrically noisy machine signals from low level control logic.

Oscillation

An effect that varies periodically between two values.

Overshoot

The amount of the parameter being controlled exceeds the desired value for a step input.

Phase-Locked Servo System

A hybrid control system in which the output of an optical tachometer is compared to a reference square wave signal to generate a system error signal proportional to both shaft velocity and position errors.

Phase Margin

The difference between 180 degrees and the phase angle of a system at the frequency where the open loop gain is unity.

PID

Proportional-Integral-Derivative. An acronym that describes the compensation structure that can be used in a closed-loop system.

PLC

Programmable Logic Controller. An industrial control device that turns on and off outputs based upon responses to inputs.

PMDC Motor

A motor consisting of a permanent magnet stator and a wound iron-core rotor. These are brush type motors and are operated by application of DC current.

Point to Point Move

A multi-axis move from one point to another where each axis is controlled independently. (No coordination between axes is required.)

Pole

A frequency at which the transfer function of a system goes to infinity.

Pole Pair, Electromechanical

The number of cycles of magnetic flux distribution in the air gap of a rotary electromechanical device.

Position Error

The difference between the present actuator (feedback) value and the desired position command for a position loop.

Position Feedback

Present actuator position as measured by a position transducer.

Power

The rate at which work is done. In motion control, Power = Torque x Speed.

Process Control

A term used to describe the control of machine or manufacturing processes, especially in continuous production environments.

Pull-In Torque

The maximum torque at which an energized stepping motor or synchronous motor will start and run in synchronism.

Pull-Out Torque

The maximum torque that can be applied to a stepping motor or synchronous motor running at constant speed without causing a loss of synchronism.

Pulse Rate

The frequency of the step pulses applied to a stepper motor driver. The pulse rate divided by the resolution of the motor/drive combination (in steps per revolution) yields the rotational speed in revolutions per second.

PWM

Pulse Width Modulation. An acronym which describes a switch-mode control technique used in amplifiers and drivers to control motor voltage and current. This control technique is used in contrast to linear control and offers the advantages of greatly improved efficiency.

Quadrature

Refers to signal characteristics of interfaces to positioning devices such as encoders or resolvers. Specifically, that property of position

Ramping

The acceleration and deceleration of a motor. May also refer to the change in frequency of the applied step pulse train.

Rated Torque

The torque producing capacity of a motor at a given speed. This is the maximum continuous torque the motor can deliver to a load and is usually specified with a torque/speed curve.

Regeneration

The action during motor braking, in which the motor acts as a generator and takes kinetic energy from the load, converts it to electrical energy, and returns it to the amplifier.

Repeatability

The degree to which the positioning accuracy for a given move performed repetitively can be duplicated.

Resolution

The smallest positioning increment that can be achieved. Frequently defined as the number of steps or feedback units required for a motor's shaft to rotate one complete revolution.

Resolver

A position transducer utilizing magnetic coupling to measure absolute shaft position over one revolution.

Resonance

The effect of a periodic driving force that causes large amplitude increases at a particular frequency. (Resonance frequency.)

RFI

Radio Frequency Interference.

Ringing

Oscillation of a system following sudden change in state.

Rise Time

The time required for a signal to rise from 10% of its final value to 90% of its final value.

RMS Current

Root mean square current. In an intermittent duty cycle application, the RMS current is equal to the value of steady state current which would produce the equivalent resistive heating over a long period of time.

RMS Torque

Root Mean Square Torque. For an intermittent duty cycle application, the RMS torque is equal to the steady state torque which would produce the same amount of motor heating over long periods of time.

Robot

A reprogrammable multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.

Robot Control

A computer-based motion control device to control the servo-axis motion of a robot.

Rotor

The rotating part of a magnetic structure. In a motor, the rotor is connected to the motor shaft.

Serial Port

A digital data communications port configured with a minimum number of signal lines. This is achieved by passing binary information signals as a time series of "1"s and "0"s on a single line.

Servo Amplifier/Servo Drive

An electronic device which produces the winding current for a servo motor. The amplifier converts a low level control signal into high voltage and current levels top produce torque in the motor.

Servo System

An automatic feedback control system for mechanical motion in which the controlled or output quantity is position, velocity, or acceleration. Servo systems are closed loop systems.

Settling Time

The time required for a step response of a system parameter to stop oscillating or ringing and reach its final value.

Shunt Resistor

A device located in a servoamplifier for controlling regenerative energy generated when braking a motor. This device dissipates or "dumps" the kinetic energy as heat.

Single Point Ground

The common connection point for signal grounds in a control wiring environment.

Slew

In motion control the portion of a move made at a constant non-zero velocity.

Slew Speed

The maximum velocity at which an encoder will be required to perform.

Speed

In motion control, the concept used to describe the linear or rotational velocity of a motor or other object in motion.

Speed Regulation

For a speed control system, speed regulation is the variation in actual speed expressed as a percentage of set speed.

SPS

Steps-Per-Second. A measure of velocity used with stepping motors.

Stall Torque

The torque available from a motor at stall or zero rpm.

Static Torque

The angle the shaft rotates upon receipt of a single step command.

Stator

The non-rotating part of a magnetic structure. In a motor the stator usually contains the mounting surface, bearings, and non-rotating windings or permanent magnets.

Stiffness

The ability to resist movement induced by an applied torque. It is often specified as a displacement curve, indicating the amount a motor shaft will rotate upon application of a known external force when stopped.

Synchronism

A motor rotating at a speed correctly corresponding to the applied step pulse frequency is said to be in synchronism. Load torques in excess of the motor's capacity (rated torque) will cause a loss of synchronism.

Tachometer

An electromagnetic feedback transducer which produces an analog voltage signal proportional to rotational velocity. Tachometers can be either brush or brushless.

Tachsyn

A brushless, electromagnetic feedback transducer which produces an analog velocity feedback signal and commutation signals for a brushless servo motor. The tachsyn is functionally equivalent to hall sensors and a tachometer.

Torque

The rotary equivalent to force. Equal to the product of the force perpendicular to the radius of motion and distance from the center of rotation to the point where the force is applied.

Torque Constant

A number representing the relationship between motor input current and motor output torque. Typically expressed in units of torque/amp.

Torque Ripple

The cyclical variation of generated torque given by the product of motor angular velocity and number of commutator segments.

Torque-to-Inertia Ratio

Defined as a motor's torque divided by the inertia of its rotor, the higher the ratio the higher the acceleration will be.

Transducer

Any device that translates a physical parameter into an electrical parameter. Tachometers and encoders are examples of transducers.

Transfer Function

The ratio of the Laplace transforms of system output signal and system input signal.

Trapezoidal Profile

A motion profile in which the velocity vs. time profile resembles a trapezoid. Characterized by constant acceleration, constant velocity, and constant deceleration.

TTL

Transistor-Transistor Logic.

Variable Frequency Drive

An electronic device used to control the speed of a standard AC induction motor. The device controls the speed by varying the frequency of the winding current used to drive the motor.

Vector Control

A method of obtaining servo type performance from an AC motor by controlling two components of motor current.

Velocity

The change in position as a function of time. Velocity has both a magnitude and a direction.

Voltage Constant (or Back EMF Constant)

A number representing the relationship between Back EMF voltage and angular velocity. Typically expressed as V/Krpm.

Zero

A frequency at which the transfer function of a system goes to zero.

INDEX

NOTE :

F PAGE NUMBER PREFIX DENOTES A FIGURE

T PAGE NUMBER PREFIX DENOTES A TABLE

- A -

A Simple Profile F3-20
ACTIVE, Area 5 3-13
Alarms (Task Levels 1-3) 4-26
  Printing with Alarms 4-27
  Restrictions of Alarms 4-27
Algebraic Functions 3-10
Allowed Tune Command
  Stability Settings T6-4
AMAX, ACC, & DEC 3-18
Analog Input 3-32
  and PDEN 4-37
Application Flowchart 4-3
Application Specification 4-3
Auto Routine (AUTO$) 4-29
Auto/Manual Flowchart F4-31
Autobauding 4-38
  Autobauding and MOTION 4-38
  Baud Rate, BAUD 4-39
  Enabling Autobaud with ABAUD 4-38
  Setting the BDS5 to Autobaud 4-38
Autobauding and MOTION 4-38
Avoiding Idling 4-26

- B -

Background (Task Level 6) 4-30
  Restrictions of Background 4-32
Backing Up the Disk(s) 2-1
Basic Commands 4-10
  BREAK (B) 4-10
  GOSUB and RET 4-11
  GOTO 4-10
  Labels 4-10
  RUN 4-10
Basic Motion Commands 3-18
  AMAX, ACC, & DEC 3-18
  EN, STOP, & LIMITS 3-18
  Enabling Motion with MOTION 3-19
  STOP (S) Command 3-19
  STOP and BREAK with
    Control X (^X) 3-19
Baud Rate, BAUD 4-39
BDS5 Conditions T4-11
BDS5 Control Modes F3-40
BDS5 Enable/Fault Logic Diagram F3-12
BDS5 Instruction Screen F2-3
BDS5 Master/Slaving F3-34
BDS5 Model Number 1-4
BDS5 Model Number Scheme F1-4
BDS5 Model Number Scheme T1-4
BDS5 Prompts T2-10
BDS5 Prompts T4-41
BDS5 Resident Editor 4-7
  CHANGE (C) 4-8
  DELETE (DEL) 4-9
  Editor Print (P) 4-7
  FIND (F) 4-8
  INSERT (I) 4-8
  NEW 4-9
  Next Line 4-7
  Password (PASS) 4-7
  Size 4-9
BDS5 Rules for Prompts T2-10
BDS5 State Table F2-11
Block-IF Restrictions and Options T4-14
BREAK (B) 4-10
Broadcast 4-41
Building A Program 4-10
  Basic Commands 4-10
  Conditional Commands 4-11

- C -

Capture 2-5
Capture Direction, CAPDIR..... 3-26
Capturing Position..... 3-26
Capture Direction, CAPDIR..... 3-26
Enabling Capture, CAP & PCAP 3-26
Speeding Up Homing Sequences 3-26
CHANGE (C)..... 4-8
Changing a Variable..... 3-3
Changing Profiles During Motion 3-30
Choosing PROTARY, PNUM, and PDEN 4-37
Clamping..... 3-27
Clamping and Homing 3-27
Clamping and Homing 3-27
Commented Program..... 4-5
Comments..... 3-1
Common User Units..... T4-32
Compensation Model Number Scheme F1-5
Compensation Module Model Number 1-5
Compliance 6-7
Computer Requirements..... 2-1
Conditional Commands..... 4-11
IF vs. ?..... 4-14
IF's with GOTO and GOSUB..... 4-15
IF, ELIF, ELSE, and ENDIF Commands..... 4-13
Nesting ? Commands..... 4-12
Nesting IF commands..... 4-15
Quick IF (?) Command..... 4-11
TIL Command..... 4-12
CONTINUE 3-37
Continuous Current, ICONT 3-17
CONTROL LOOPS 3-37
Position Loop 3-38
Power-Up Control Loops 3-39
Torque Command..... 3-39
Velocity Loop..... 3-38
Controlling the Velocity Loop with PROP..... 3-16
Critical Damping..... 6-2
Critical Damping..... F6-2
Current 3-16
Current Limits, IMAX & ILIM 3-16
Motor Current, ICMD & IMON..... 3-16
Current Limits, IMAX & ILIM 3-16
Current Units..... 4-32
Cursor 2-8
Cursor Addressing..... 4-19
Cursor Control Keys..... T2-8
Customer Service 4-6

- D -

Debugging and Multi-Tasking..... 5-2
Debugging Modes 5-1
Single-Step 5-1
Trace 5-2
DELETE (DEL) 4-9
DEP 5-8
Descriptions of Modes 2-10
Interactive Mode 2-10
Monitor Mode 2-12
Other Modes..... 2-13
Run Mode..... 2-12
Single-Step Mode..... 2-12
Trace Mode 2-12
Desired Operation of Program Example T4-14
Direction Control, DIR..... 3-14
Displaying Error Messages 5-9
Drive Control 3-14
Controlling the Velocity Loop with PROP..... 3-16
Current 3-16
Direction Control, DIR..... 3-14
Enabling the BDS5..... 3-16
Enabling the Position Loop with PL..... 3-16
Limiting Motor Current..... 3-17
Position 3-14
Velocity..... 3-15
DWELL (D) 4-22

- E -

Edit 2-7
Editing 4-6
BDS5 Resident Editor 4-7
Motion Link Editor 4-6
Editor 2-7
Cursor..... 2-8
Edit..... 2-7
File 2-7
GOTO 2-8
Help..... 2-8
Insert/Delete 2-8
Editor Print (P)..... 4-7
Electronic Gearbox..... 3-32
Gear Ratio, GEARI & GEARO..... 3-32
Gearbox Example 1..... 3-32
Gearbox Example 2..... 3-33
Gearbox, ACC/DEC, and Jogs 3-35
Profiles and Gearbox..... 3-33
Velocity Offset, VOFF 3-35
EN, STOP, & LIMITS 3-18
Enabling and Disabling Multi-tasking..... 4-23
Enabling Autobaud with ABAUD..... 4-38
Enabling Capture, CAP & PCAP 3-26
Enabling Motion with MOTION..... 3-19

Enabling the BDS5 3-16
Enabling the Position Loop with PL 3-16
Encoder Feedback..... 3-37
Encoder Resolution.....T3-37
END Command..... 4-23
English Conversion (12-bit
R/D Only) T4-35
Environmental Specifications T1-12
ER-External Resistor Kit Model Number 1-7
Error Handler (ERROR$) 4-29
Error History..... 5-9
Error Levels 5-8
Error Log 5-8
DEP 5-8
Displaying Error Messages 5-9
Error History..... 5-9
Error Levels 5-8
Firmware Errors..... 5-9
Error Severity Levels and Actions T5-8
Establishing Communications..... 2-3
Example Application..... 4-3
External Inputs..... 3-31
Analog Input 3-32
External Regen Resistor Model
Number Scheme..... F1-7
External Regen Resistor Model
Number Scheme.....T1-7
External Units 4-33
External Units Conversion T4-36

- F -

Fault Latch, Area 3 3-13
Fault Logic..... 3-11
ACTIVE, Area 5..... 3-13
Fault Latch, Area 3 3-13
Fault Logic, Area 2 3-13
Firmware Faults, Area 1..... 3-13
Motor Brake..... 3-14
Output Relay..... 3-14
Ready Latch, Area 4..... 3-13
Relay and STATUS Control, Area 6..... 3-13
Fault Logic, Area 2 3-13
Features 1-1
File 2-7
FIND (F) 4-8
Firmware Errors..... 5-9
Firmware Faults, Area 1..... 3-13
Foldback Current, IFOLD..... 3-17
Four Idling Commands T4-26

- G -

Gating Motion with GATE 5-6
Gear Ratio, GEARI & GEARO 3-32
Gearbox Example 1..... 3-32
Gearbox Example 2..... 3-33
Gearbox, ACC/DEC, and Jogs..... 3-35
General Purpose Input/ Output 3-10
Whole Word I/O 3-10
GOSUB and RET..... 4-11
GOTO 2-8
GOTO 4-10

- H -

Hardware Travel Limits 3-19
Help 2-6
Help 2-8
Hexadecimal 3-8
Hints 5-6
HOLD (H)..... 4-21
How to Disable Multitasking T4-25
How to Enable Multi-Tasking.....T4-25

- I -

Idling 4-25
Avoiding Idling..... 4-26
Post-Execution Idle..... 4-26
Pre-Execution Idle 4-25
Idling Commands..... 4-21
DWELL (D)..... 4-22
HOLD (H)..... 4-21
WAIT (W) 4-22
IF vs. ? 4-14
If Your System Is Completely Unstable..... 6-3
IF's with GOTO and GOSUB 4-15
IF, ELIF, ELSE, and ENDIF
Commands 4-13
Incremental Move Example 3-22
Indirect User Variables 3-7
Initial Settings of Control
and User Variables..... 3-4
INPUT 4-20
INPUT and Decimal Point..... 4-21
INPUT Limits 4-20
Input 1-16 Decimal Values T3-11
INPUT and Decimal Point..... 4-21
INPUT Limits 4-20
INSERT (I) 4-8
Insert/Delete..... 2-8
Install on a Floppy Disk..... 2-2
Install on a Hard Disk 2-2
Instructions..... 3-1
Comments 3-1

Integrating Velocity Loop 3-38
Interactive Mode 2-10
Interfacing with the Operator 4-16
INPUT 4-20
PRINT (P) 4-17
REFRESH (R & RS) Commands 4-20
SERIAL Switch 4-21

- J -

JOG (J) Command 3-23
Jog From (JF) Command F3-28
JOG TO (JT) & JOG FROM (JF) 3-28
Changing Profiles During Motion A3-30
Multiple JF/JT Commands 3-30
Registration A3-29
Registration Example 3-29
Jog To (JT) Command F3-29

- K -

- L -

Labels 4-10
Limiting Motion 3-19
Hardware Travel Limits 3-19
Software Travel Limits, PMAX & PMIN 3-20
User Position Trip Points, PTRIP1 & PTRIP2 3-20
Limiting Motor Current 3-17
Continuous Current, ICONT 3-17
Foldback Current, IFOLD 3-17
Monitoring Current Limits 3-18
Logical Functions: AND, OR 3-9
Low-Pass Filters 6-8

- M -

Machine Specific Units 4-35
Macro Move Example #1 3-25
Macro Move Example #1 ..... F3-25
Macro Move Example #2 3-25
Macro Move Example #2 ..... F3-25

MACRO MOVES 3-24
Macro Move Example #1 3-25
Macro Move Example #2 3-25
MCA, MCI, MCD, & MCGO 3-24
Main Program Level (Task Level 5) 4-29
Auto Routine (AUTO$) 4-29
Error Handler (ERROR$) 4-29
Manual Program (MANUAL $) 4-30
Power-Up Routine (POWER-UP$) 4-29
Typical AUTO/MANUAL Programs 4-30
Manual Program (MANUAL $) 4-30
Master/Slave Block Diagram F4-34
Math 3-8
Algebraic Functions 3-10
Hexadecimal 3-8
Logical Functions: AND, OR 3-9
MCA, MCI, MCD, & MCGO 3-24
Mechanical Specifications T1-12
Menus and Windows 2-4
Capture 2-5
Help 2-6
Options 2-5
Program 2-4
Scope 2-5
Utilities 2-6
Variables 2-4
Metric Conversion (12-bit R/D Only) T4-35
Molex Assembly Tools 1-7
Monitor Mode 2-12
Monitor Mode Commands T2-12
Monitoring Current Limits 3-18
Motion Commands 3-18
Basic Motion Commands 3-18
Capturing Position 3-26
Clamping 3-27
CONTINUE 3-37
Electronic Gearbox 3-32
Encoder Feedback 3-37
External Inputs 3-31
JOG (J) Command 3-23
JOG TO (JT) & JOG FROM (JF) 3-28
Limiting Motion 3-19
MACRO MOVES 3-24
NORMALIZE (NORM) Command 3-23
Profile Regulation 3-35
Profiles 3-20
R/D BASED MOVE (MRD) Command 3-26
Zero Position Error (ZPE) Command 3-24
Motion Link and Trace 5-2
Motion Link Editor 4-6

Motion Link Overview 2-4
Menus and Windows 2-4
Editor 2-7
Types Of Data Files 2-9
Using IBM-PC Compatibles 2-9
Motion Link Setup Program 2-9
Motion Segments 5-4
Motor Brake 3-14

Motor Current, ICMD & IMON 3-16
Move Absolute (MA) Command 3-21
Move Incremental (MI) Command 3-22
MULTI 4-23
MULTI-TASKING 4-23
Alarms (Task Levels 1-3) 4-26
Background (Task Level 6) 4-30
Enabling and Disabling Multi-tasking 4-23
END Command 4-23
Idling 4-25
Main Program Level (Task Level 5) 4-29
MULTI 4-23
Multitasking and Autobauding 4-23
Variable Input (Task Level 4) 4-27
Multi-Tasking Debug Prompts T5-3
Multi-Tasking Overview T4-24
Multidrop Communications 4-40
Broadcast 4-41
Multiple JF/JT Commands 3-30
Multiple Profile Commands 3-23
Multitasking and Autobauding 4-23

- N -

Nesting ? Commands 4-12
Nesting IF commands 4-15
NEW 4-9
Next Line 4-7
Non-Linear Mechanics 6-7
NORMALIZE (NORM) Command 3-23

- O -

Options 2-5
Other Modes 2-13
Other User Units 4-33
Output 1-8 Decimal Values T3-10
Output Relay 3-14
Overdamping 6-2
Overdamping F6-2
Overloading the Motor 6-6

- P -

Part Number Description 1-3
BDS5 Model Number 1-4
Compensation Module Model Number 1-5
ER-External Resistor Kit Model Number 1-7
Molex Assembly Tools 1-7
PSR4/5 Model Number 1-6
Password (PASS) 4-7
Position 3-14
Position Command and Feedback, PCMD & PFB 3-14
Position Error, PE & PEMAX 3-14
R/D Position, PRD 3-15
Sampling PFB, PCMD and PEXT 3-15
Position Command and Feedback, PCMD & PFB 3-14
Position Error, PE & PEMAX 3-14
Position Loop 3-38
Position Rotary Mode, ROTARY, & PROTARY 4-37
Choosing PROTARY, PNUM, and PDEN 4-37
Rotary Mode and Absolute Moves 4-38
Post-Execution Idle 4-26
Power-up and Control Variables 3-3
Power-Up Control Loops 3-39
Power-Up Routine (POWER-UP$) 4-29
Power-Up State of Programmable Units T3-4
PRD: Ranges and R/D Resolutions T3-15
Pre-Execution Idle 4-25
PRINT (P) 4-17
Cursor Addressing 4-19
Printing ASCII Characters 4-18
Printing BDS5 Status (PS) 4-20
Printing Binary Numbers 4-18
Printing Control Characters 4-19
Printing Decimal Numbers 4-17
Printing Decimal Points 4-17
Printing Expressions 4-18
Printing Hex Numbers 4-17
Printing Switches 4-18
Printing ASCII Characters 4-18
Printing BDS5 Status T4-20
Printing BDS5 Status (PS) 4-20
Printing Binary Numbers 4-18
Printing Control Characters 4-19
Printing Decimal Numbers 4-17
Printing Decimal Points 4-17
Printing Expressions 4-18
Printing Hex Numbers 4-17
Printing Switches 4-18
Printing Variables 3-2
Printing with Alarms 4-27
Problems 6-6
Compliance 6-7
Low Pass Filters 6-7
Non-Linear Mechanics 6-7
Overloading the Motor 6-6
Resonance 6-7

Processor Modes 2-9
Descriptions of Modes 2-10
Prompts 2-9
Product Description..... 1-1
Profile Final Position, PFNL..... 3-23
Profile Limits 3-22
Profile Regulation 3-35
Profile Regulation and Counting Backwards..... 3-36
REG & REGKHZ..... 3-35
Regulation Example 3-36
Profile Regulation and Counting Backwards..... 3-36
Profiles 3-20
Incremental Move Example 3-22
Move Absolute (MA) Command..... 3-21
Move Incremental (MI) Command..... 3-22
Multiple Profile Commands 3-23
Profile Final Position, PFNL..... 3-23
Profile Limits 3-22
S-Curves..... 3-20
Profiles and Gearbox..... 3-33
Program 2-4
Program Examples 4-41
Programming Conditions..... 3-3
Programming Techniques..... 4-1
Application Flowchart..... 4-3
Application Specification..... 4-3
Commented Program..... 4-5
Customer Service 4-6
Example Application..... 4-3
Prompts 2-9
Prompts 4-39
Proportional Velocity Loop..... 3-38
PROTARY 4-37
PSR4/5 Model Number..... 1-6
PSR4/5 Model Number Scheme..... F1-6
PSR4/5 Model Number Scheme..... T1-6

- Q -

Quick IF (?) Command..... 4-11

- R -

R/D BASED MOVE (MRD) Command 3-26
R/D Converter Accuracy..... T3-26
R/D Position, PRD 3-15
Ready Latch, Area 4..... 3-13
RECORD and PLAY 6-6
Reducing ILIM..... 6-3
REFRESH (R & RS) Commands 4-20
REG & REGKHZ..... 3-35
Registration 3-29
Registration Example 3-29
Regulation Example 3-36
Regulation Timer, RD..... 5-4
Relay and STATUS Control, Area 6..... 3-13
Removing Code..... 5-3
Resonance 6-7
Restrictions of Alarms..... 4-27
Restrictions of Background..... 4-32
Restrictions of Variable Input 4-28
Ringing 6-2
Ringing F6-2
Rotary Mode and Absolute Moves..... 4-38
Rules For Math Expressions..... T3-9
RUN 4-10
Run Mode..... 2-12

- S -

S-Curve Acceleration Chart T3-21
S-Curve Profile..... F3-21
S-Curves 3-20
Sample Flowchart..... F4-4
Sampling PFB, PCMD and PEXT 3-15
Scope 2-5
Segments for Different Moves T5-5
Serial Communications 4-38
Autobauding..... 4-38
Multidrop Communications..... 4-40
Prompts 4-39
Serial Watchdog..... 4-39
System Dump 4-40
Transmit/Receive Programs 4-39
SERIAL Switch..... 4-21
Serial Watchdog..... 4-39
Setting External Units in Master/Slave Systems..... T4-33
Setting the BDS5 to Autobaud 4-38
Simplified Schematic Diagram and System Diagram..... 1-13
Single-Step 5-1
Single-Step Mode..... 2-12
Size 4-9
Software Installation..... 2-1
Software Installation..... 2-1
Backing Up the Disk(s) 2-1
Establishing Communications 2-3
Software Installation..... 2-2
Software Installation..... 2-2
Install on a Floppy Disk 2-2
Install on a Hard Disk..... 2-2
Software Travel Limits, PMAX & PMIN 3-20
Special Constants 3-8
Specifications T1-8

Specifications and Ratings 1-8
Speeding Up Homing Sequences 3-26
Standard Units..... T3-2
STOP (S) Command 3-19
STOP and BREAK with
Control X (^X)..... 3-19
Switche s..... 3-2
Synchronizing Your Program..... 5-4
Gating Motion with GATE 5-6
Motion Segments 5-4
Regulation Timer, RD..... 5-4
Using the Timers, TMR1-4..... 5-4
WAIT (W) 5-5
System Compensation..... 6-1
Critical Damping..... 6-2
Overdamping..... 6-2
Ringing..... 6-2
Underdamping..... 6-2
System Dump..... 4-40
Version Dump..... 4-40
System Resolutions T4-33

- T -

The >BDS Command Transmitting
to the BDS5..... 4-39
Theory of Operation..... 1-12
Three Types of Variables..... 3-2
TIL Command..... 4-12
To Execute AUTO$ T4-30
To Execute MANUAL$..... T4-30
Torque Command 3-39
Trace 5-2
Motion Link and Trace 5-2
Trace Mode..... 2-12
Transmit/Receive Programs 4-39
<BDS Command Receiving from
the BDS5..... 4-39
The >BDS Command Transmitting
to the BDS5..... 4-39
TUNE Command 6-4
Tuning 6-3
If Your System Is Completely Unstable..... 6-3
Reducing ILIM..... 6-3
Tuning Criterion..... T6-1
Tuning the BDS5 Yourself 6-4
Tuning the Position Loop..... 6-5
Tuning the Velocity Loop 6-4
Tuning the Position Loop..... 6-5
Tuning the Velocity Loop 6-4
Types Of Data Files 2-9
Typical AUTO/MANUAL Programs..... 4-30

- U -

Underdamping..... 6-2
Underdamping..... F6-2
Units 4-32
Machine Specific Units..... 4-35
Position Rotary Mode, ROTARY, &
PROTARY..... 4-37
User Position Trip Points, PTRIP1
& PTRIP2 3-20
User Switches..... 3-8
User Units 4-32
Current Units..... 4-32
External Units 4-33
Other User Units 4-33
User Variables..... 3-7
Indirect User Variables 3-7
Using IBM-PC Compatibles 2-9
Using the General Purpose Inputs..... 4-15
Using the Timers, TMR1-4..... 5-4
Using Variable Input with Profiles..... 4-28
Utilities 2-6

- V -

Variable Input (Task Level 4)..... 4-27
Restrictions of Variable Input..... 4-28
Using Variable Input with Profiles..... 4-28
Variable Limits 3-2
Variable Units..... 3-2
Variables..... 2-4
Variables..... 3-1
Changing a Variable..... 3-3
Initial Settings of Control
and User Variables..... 3-4
Power-up and Control Variables..... 3-3
Printing Variables 3-2
Programming Conditions 3-3
Special Constants..... 3-8
Switches..... 3-2
Three Types of Variables..... 3-2
User Switches..... 3-8
User Variables..... 3-7
Variable Limits 3-2
Variable Units..... 3-2
VCMD, VFB, VE, & VAVG..... 3-15
Velocity 3-15
VCMD, VFB, VE, & VAVG..... 3-15
Velocity Limits, VMAX & VOSPD 3-16
Velocity Limits, VMAX & VOSPD 3-16
Velocity Loop 3-38
Integrating Velocity Loop..... 3-38
Proportional Velocity Loop 3-38

Velocity Loop Bandwidth vs. KP_{MAX} T6-5
Velocity Loop Bandwidth vs. KVI.....T6-5
Velocity Offset, VOFF 3-35
Version Dump 4-40

- W -

WAIT (W).....4-22
WAIT (W).....5-5
Whole Word I/O.....3-10

- X -

- Y -

- Z -

Zero Position Error (ZPE) Command.....3-24

- - -

<BDS Command Receiving from
the BDS5.....4-39

BDS5 UPGRADE NOTICES

VERSION 3.0.x FIRMWARE HISTORY

DATE	LLL	FIRMWARE VERSION	OBSOLETE/ CURRENT/ S.O.	MC VERSION	DESCRIPTION OF CHANGES
10-93	030	3.0.0	Obsolete	MC2 Rev 2 and later	• Added "MONITOR" switch to force Monitor mode during program run. • Added a serial port "ECHO" switch. • Added "MSG" switch to control the Power-up and Monitor messages. • Added a fix for the Serial watchdog (took 35 sec to print). • Added a random number command ("RAND"). • Added Print Append command ("PA and "PAS"). • Changed the "RUN" command to work from user program • Added Real Time Trace commands ("TRECORD" and "TPLAY"). • Added an Electronic Cam mode (CAM and PCAM). • Enhanced "NORM" command to enable Camming • Moved position interrupt code around to make CAM work properly. This reduced the loop delay and increased BDS5 stability (i.e. bandwidth). • Stopped "CONTINUE" from working with CAM. • Made drive disable also disable Camming. • Added "LPF" and "LPFHZ" to the "DUMP TL" command. • Added Extended User Registers "EXTDX" for X251-X750. • Added "CLEARX" command. • Fixed the 16-bit R/D problem. • Fixed motion bug due to multiple "J 0" commands • Eliminated micro register test to make more room for code. • Fixed bug that ignored Escape when background was continuously printing. • Fixed bug where error messages occasionally printed the wrong line number. • Added code to initialize the new Dallas Bat-RAMS to prevent power-up and program crash problems. • Register based instructions will execute .3 msec slower.

VERSION 3.0.x FIRMWARE HISTORY (Continued)

DATE	LLL	FIRMWARE VERSION	OBSOLETE/ CURRENT/ S.O.	MC VERSION	DESCRIPTION OF CHANGES
10-93	030	3.0.1	Obsolete	MC2 Rev 2 and later	• Changed the lower limit of "PEMAX" from 0 to 1. • Force "PEMAX" to 1 if less than 1.
2-94	030	3.0.2b	Obsolete Beta Bug Fix (Shipped to some customers for test and verification of bug fixes)	MC2 Rev 2 and later	• Enhanced the Message command ("MSG") to also suppress error messages. • Fixed error handler to properly handle more than one error. • Fixed ERROR$ to reliably execute to completion on first error. • An error will now force off Monitor mode to prevent program hang. • Fixed Jog command ("J") that could cause software watchdog if executed during an error.
2-94	030	3.0.2	Obsolete	MC2 Rev 2 and later	• Functionally equivalent to BDS5 3.0.2b plus. . . • Current foldback fix • INPUT"" command fix • Clear "LSTERR" to 0 on "EN". • Fixed Print problem due to program terminating with an "END" while in Monitor mode. KNOWN PROBLEMS • On power-up the program corrupt message is suppressed. • The TPLAY command will not display the newest trace line if the number of lines stored exceed 1000. • On power-up with a new uninitialized BATRAM the BDS5 will often trigger a software watchdog (this is an inconvenience to the factory test people).
4-94	030	3.0.3	Obsolete	MC2 Rev 2 and later	• Fixed non-printing error message on power-up due to corrupt user program. • Fixed TPLAY to properly display the trace information after the number of lines stored exceed 1000. • Fixed the factory initialization problem that would cause a software watchdog trip due to the user program being uninitialized in a new BATRAM chip.

VERSION 3.0.x FIRMWARE HISTORY (Continued)

DATE	LLL	FIRMWARE VERSION	OBSOLETE/ CURRENT/ S.O.	MC VERSION	DESCRIPTION OF CHANGES
5-94	030	3.0.4	Obsolete	MC2 Rev 2 and later	• Enhanced the CLEARX command to allow separate clearing of all User Registers or User Flags or both. • Fixed LSTERR to work even if MSG=0. • Fixed another factory initialization problem that would cause a software watchdog trip on entry to the User Program Editor due to the user program being uninitialized in a new BATRAM chip. • Enhanced the Break (B) command to allow the optional breaking of a user INPUT command (B I). • Enhanced the Error Recovery Task to cancel the user INPUT command if it is active. • Enhanced the Error Recovery Task to cancel a Task idling command such as “H”.
11-94	030	3.0.5	**Current**	MC2 Rev 2 and later	• Added MONITOR=OFF support from the user program. • Fixed a problem with EXTLOOP. • Added PCMD=PEXT when EXTLOOP=1 and drive disabled. • Fixed front panel LED initialization on power-up. • Syntax error in ERROR$ will break the program. • An invalid command will not allow any program execution. • An error occurring while in Monitor mode will print the error message and return to Monitor mode.

BDS5 UPGRADE NOTICE

VERSION 3.0.0

- It provides many new commands to control serial communications with.
- It provides the user with up to 750 user registers.
- It provides a real time trace function.
- It provides an electronic cam function.
- It provides a random number generator.
- It provides for using the run command from within the user program for purposes of restarting a program.
- It provides up to 5% higher position loop bandwidth.

EXISTING APPLICATIONS

Please be aware that register based operations will run approximately .3 msecs slower (see next page). This could cause a problem with older, time critical applications. Should this problem occur with older applications - please contact the factory for the older firmware version 2.0.6.

UPGRADES

When upgrading older systems with 3.0.0. be sure to initialize the three new non-volatile flags for proper operation:

- ECHO=1
- MSG=1
- EXTDX=0

750 USER REGISTERS

Be aware that while the extra 500 user registers are enabled, the PC-SCOPE and PC-TRACE commands will be unavailable.

MONITOR 0|1	This will automatically force entry into the Monitor mode at the start of running a program. MONITOR is set to 0 on Power-up.
ECHO 0|1	Used to suppress the echo of serial port characters ECHO is remembered on Power-up.
MSG 0|1	Used to suppress the power-up message and the Monitor mode message. MSG is remembered on Power-up.
PA <expr>{[<format>]} | "<text>" {...}	Print Append command - This command is like the Print command except there is no terminating <carriage return> & <linefeed>. Also see the "R" and "RS" print command.
PAS <expr>{[<format>]} | "<text>" {...}	Print Append Status command - This is identical to the PA command except it also prints drive status information.
RUN {<label>}	Now Allowed from within program - This can allow the user to "restart" a program from an Alarm or Error label.
CLEARX	This command will clear all user registers and user flags to zero. Note: The Variable upload can be greatly sped up if all user registers and flags are zero. Edit the .VAR file and remove all XS1-XS50 ad X1-X750 lines and replace them with one "CLEARX" command. (X1-X750 & XS1-XS50
EXTDX 0|1	This will enable the Extended User Registers X251-X750. Note: The PC-SCOPE command and the Real Time Trace command can not be used while the Extended User Registers are enabled. EXTDX is remembered on Power-up
RAND <seed> RAND X<user register> <expr1> <expr2>	Random number generator command <seed> = 0 - 31,000 -This generates the random number sequence. <user reg>=1-250 -This is the user register used to store the random number. <expr1> -This is a random number boundary (limit). <expr2> -This is the other random number boundary (limit). |<expr1>-<expr2>| <=31,000
CAM	This internal switch indicates if the 128 point Camming mode with linear interpolation is enabled.
PCAM	Position command from CAM table. The CAM table is contained in user registers X100-X227.

NORM <pos> {CAM}	Used to enable the Cam at the Normalize position - GEAR mode must be turned off before enabling CAM, after which GEAR must be turned on to start Camming.
DUMP {TL | VERSION}	Added LPF and LPFHZ to DUMP TL

TRECORD START | STOP | CONTINUE	Real Time Trace Record command. This can be buried in the user program to control Real Time Tracing.
TPLAY TPLAY NEWEST <#lines> TPLAY OLDEST <#lines> TPLAY <#lines> TPLAY <line1> <line2>	Real Time Trace Playback command. - this will dump the entire Real Time Trace buffer to the serial port. -this will output the newest # of lines. -this will output the oldest # of lines. -this will output a # of lines. -this will output from <line1> to <line2>. See the Output syntax below

Output Format: #<line> LINE <pgm line> <prompt> <command>

#<line> is the trace line number.

<pgm line> is the user program line number.

<prompt> is a modified trace prompt that in addition to indicating the trace mode and the multi-drop address, also indicates the multi-task active.

- ta. indicates Alarm A$ was active
- tB. indicates Alarm B$ was active
- tC. indicates Alarm C$ was active
- tV. indicates Variable$ was active
- t.. indicates the MAIN PROGRAM ($-$500) was active
- t*. indicates BACKGROUND$ was active
- tI. indicates INTERACTIVE was active
- tM. indicates MONITOR was active.

<command> is the user program command that executed.

```
Output Example:  #1 LINE 22  AT .. 5$      ;this indicates program trace line1 was
                                           ;at user program line 22 from the Main
                                           ;program task and it was a label (5$).
```

NOTE: When upgrading from an older version of firmware the following variable must be initialized to insure proper operation of the BDS5. This is because they are not forced to default power-up state and therefore will assume the value of the previously unitialized memory locations. the flags ECHO and MSG should be set to 1 and the flag EXTDX should be set to 0 for 250 user registers and set to 1 for 750 user registers. Remember that when EXTDX is enabled, it will disable the PC-SCOPE and PC-TRACE features of the BDS5.

- ECHO=1
- MSG=1
- EXTDX=0

BDS5 TIMING TESTS

INSTRUCTION	SPECIAL CONDITION	TIMING METHOD	TIME (msec) Ver 2.0.5	TIME (msec) Ver 2.0.5
O1 ON	None	1	1.90	1.94
O1 OFF	None	1	1.90	1.92
OUT=OUT!0C8h	None	1	2.86	2.89
? 1 EQ 1 01 ON	None	1	3.99	4.04
X1=2	None	1	1.72	2.03
X1=X2+1	None	1	2.18	2.50
X1=X2-1	None	1	2.17	2.50
X1=X2*100	None	1	2.33	2.65
X1=X2/100	None	1	2.36	2.67
ZPE	None	1	1.20	1.67
NORM 0	None	1	2.00	2.64
O1 On	GEAR ON	1	2.07	2.11
X1=X2+1	GEAR ON	1	2.37	2.72
MI 40960 1000	None	2	6.00	6.00
MI 40960 1000	GEAR ON	2	6.80	6.80
MI 40960 1000	GEAR & REG ON	2	8.00	8.00
MI 40960 1000	REG ON	2	6.20	6.10
MI 40960 1000	REG ON & IN MOTION	2	7.00	7.00
MI 40960 *	NONE	2	5.00	5.10
MI 40960 *	GEAR ON	2	5.80	5.40
MI 40960 *	REG ON	2	5.10	5.20

* (VDEFAULT 1000)

TIMING METHOD #1	TIMING METHOD #2
Divide the number of loops counted (X1) into 10 seconds (10 / X1) to get the single loop time. Perform this operation twice, once with the <instruction> and once without the <instruction>. Subtract the amount of time it takes to run this program with the <instruction> from the time to run without the <instruction>. The result is the amount of time it takes to execute the instruction.	With an oscilloscope measure the amount of time that Output #1 is ON while running the program with the <instruction> and subtract from this the amount of time the Output #1 is ON while running the program without the <instruction>. The result is the amount of time it takes to execute the instruction.
4$ X1=0 TMR1=1---- 5$ X1=X1+1 <instruction> ? TMR1 GT 0 GOTO 5 B	5$ PLIM 0 EN O1 ON <instruction> O1 OFF W0 D 1000 GOTO 5

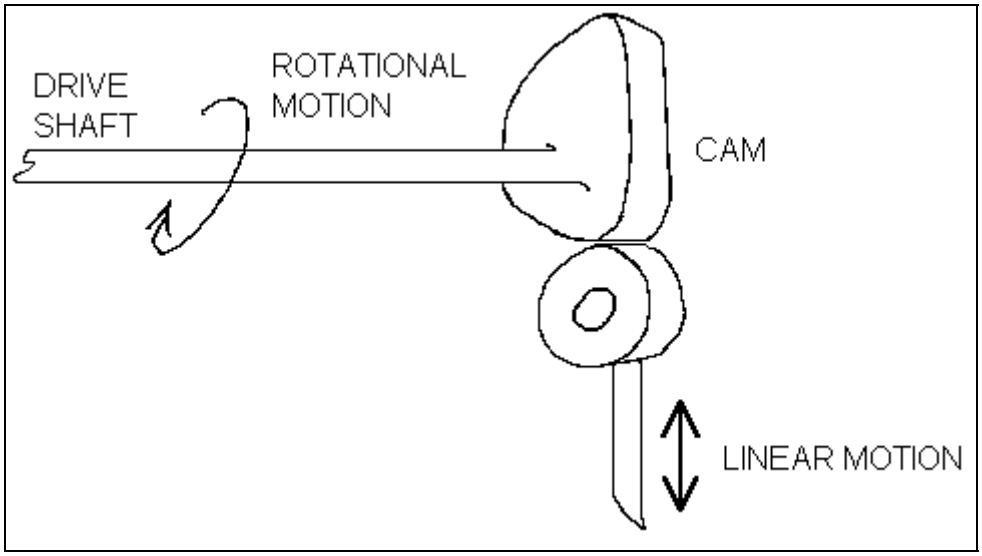


Figure 1. A Conventional Cam

Conventional Cams

Conventional cams convert rotational motion to linear motion. As Figure 1 shows, the "command" signal comes from a master-drive shaft which is fitted with a cam. The cam generates linear motion on a follower. Cams are used when a specific profile must be generated each time a drive shaft turns. The cam itself can have a wide variety of shapes. This leads to tone of the most important features of cams: you can generate a wide range of linear motion profiles using only constant rotational motion.

Electronic cams offer many advantages over conventional, mechanical cams. For example, the profile of an electronic cam is much easier to change. These profiles can be changed without machining parts, and without disassembling the machine. With electronic cams, the machine comes off-line only for the few minutes it takes to load a new profile. another advantage is that electronic cam profile are not subject to wear like their mechanical counterparts.

The profile of an electronic cam is stored in a table such a the one shown in Figure 2. This table defines the relationship between the drive and follower positions. You define the table directly or take it from an existing conventional cam. If the cam already exists, you determine the radius of the cam profile at different angles as shown in Figure 3. Here, 4 radii are shown, though in practice, many more are specified.

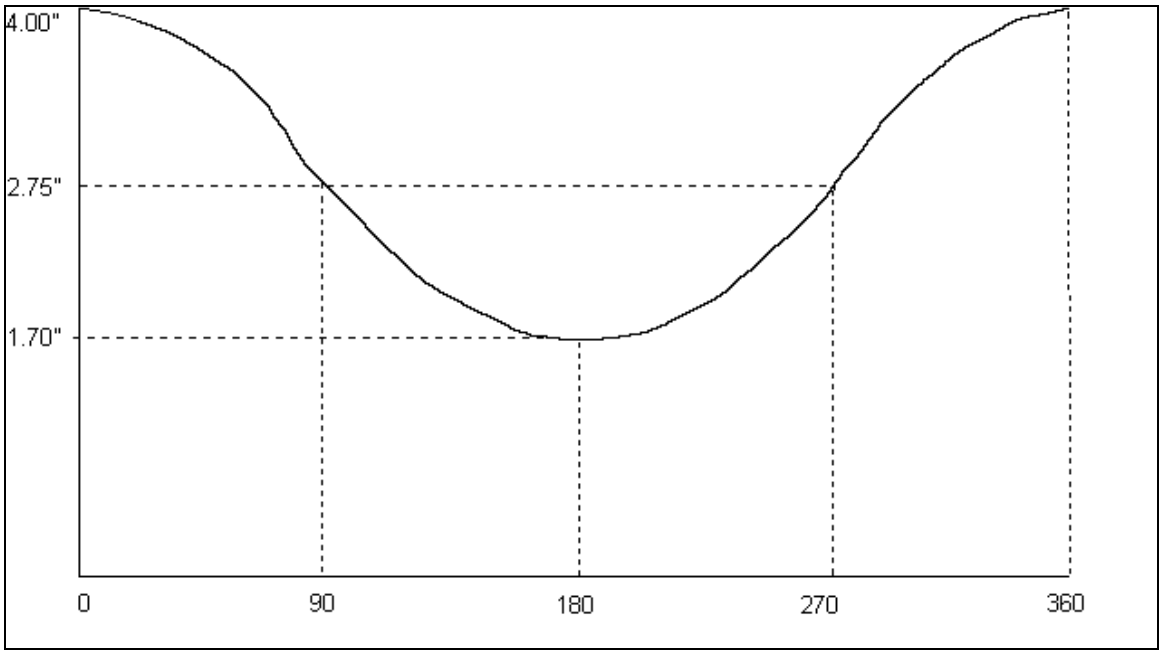


Figure 2. Electronic Cam Table

One of the most important features of an electronic cam is that the master drive can rotate in one direction indefinitely. With conventional positioners, this will eventually cause an error because the internal position counter will overflow. Also, the electronic cam controller must support gear ratios between the drive shaft and the follower. Again, the drive can rotate indefinitely, and the controller must not lose counts. The BDS5 has been designed with both of these criteria, allowing it to serve as both a conventional positioner and as an electronic cam.

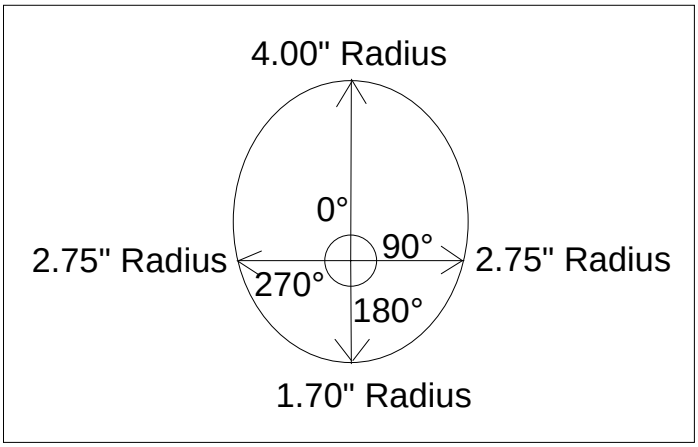


Figure 3. Conventional Cam Table

Setting up the BDS5

To use BDS5 camming, you need to follow these steps:

1) Generate a cam table and enter it into the BDS5.
2) Scale the BDS5 electronic gearbox.
3) Align the machine and enable camming.

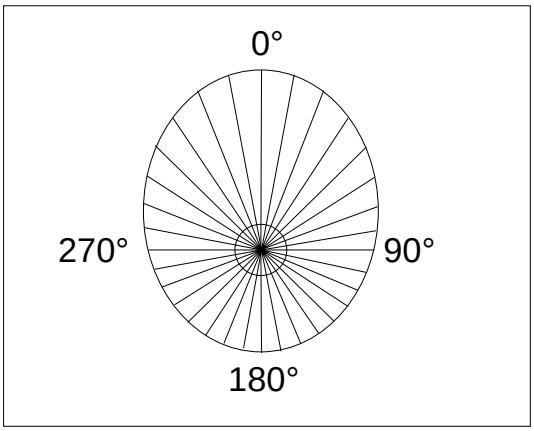


Figure 4
Dividing a CAM

1) Generating a Table

To generate a table, start with a graph, like the one in Figure 2, showing the master drive position versus the follower position. Divide this graph into 128 evenly spaced sections. Each section represents about 2.81 degrees (360/128). Now load the follower positions into the BDS5 user variables X100-X227 as shown below:

Master Position	Master Degrees	Load Follower Position in This User Variable
1	0	X100
2	2.81	X101
3	5.62	X102
4	8.43	X103
.	.	.
.	.	.
.	.	.
126	351.56	X225
127	354.38	X226
128	357.19	X227

Note that the beginning position (X100) should be close to the ending position (X227). This is because this is an absolute electronic cam that will always cycle back to it's original starting position. and begin the cam again. The next position after X227 is X100. when the cam table is written it must wrap around so that position X227 and position X100 are close. If the positions are not close the motor will jump and will possible trip out due to either an overspeed or a position following error.

2) Scale the gearbox

The BDS5 processes the master drive signal through the gearbox so you can select the gear ratio you need. You must select the gear ratio so that when the master rotates 360 degrees, 32,768 counts are generated in the BDS5. think of the 360 degree horizontal axis of Figure 2 as being 32.768 counts long. for example, suppose the sensor on the drive shaft were a 1000-line encoder. Because of quadrature, the encoder would generate 4000 counts for every rotation. So the 4000 counts should be scaled through the gearbox to generate 32,768 counts. The gear ratio would be:

$$\frac{\text{GEARI}}{\text{GEARO}} = \frac{32,768}{4000} = \frac{4096}{500}$$

or, GEARI=4096 and GEARO=500.

If you want to test your scaling, enable your BDS5 (without camming), turn GEAR ON, and rotate the drive motor 360 degrees. The follower motor should rotate 32,768 counts (8 rev's for a 12-bit system or 2 rev's for a 14-bit system).

3) Align the Machine

Some applications require that the master drive be aligned; others assume the drive is 0 degrees at power-up. If your application requires drive shaft alignment, you must provide the necessary mechanisms as the BDS5 will not have control of the drive shaft. Virtually all applications require that you align, or "home", the follower position. Depending on your system, you may need a home switch. In any event, the two positions must line-up somewhere in the cam table. For example, in Figure 2, if the drive shaft were at 0 degrees and the follower were at 3.00 inches, either the master drive or the follower would have to move. You would either have to:

1) rotate the drive to 90 degrees to line up with the 3.00 inch follower position, or
2) move the follower to 4.00 inches to line up with the 0 degree master-drive position.

The master and follower must line up somewhere in the cam table before you can proceed.

The same command both enables camming and aligns the master and follower. The BDS5 uses the NORM command with "CAM" as the third entry. The form of the command is:

NORM <Master Drive Position> CAM

This command simultaneously aligns and enable camming. When camming is enabled, the software switch CAM is on. To disable camming, you must reset the BDS5, disable the BDS5, or enter the NORM command without the "CAM". You cannot directly change the value of CAM. Note that the BDS5 must be enabled to use NORM with the "CAM" entry.

When the "NORM <Master Drive Position> CAM" command is executed, PCMD is determined by the cam table and is set to the corresponding value for PCMD. PFB is set to the same value as PCAM and therefore there is no PE (position error).

Returning to Figure 2, assume you know the follower to bed 4.00 inches. The master drive should be at 0 degrees. You would normalize the position as follows:

```
EN
NORM 0 CAM
```

If you are using position units, the most convenient place to normalize for camming is when the master drive is at zero. This is because the master drive position (PCMD) uses position units (PNUM and PDEN) which are normally scaled for the follower. When you normalize to zero, the units do not have any effect. However, if you want to normalize the master to a non-zero position, you must

A) Determine the position of the drive master to which you will normalize;

B) Convert the position to counts where 360 degrees equals 32.768 counts;

C) Temporarily set position units to 1:1 (PNUM = PDEN = 1);

D) Normalize to the position in counts:

```
EN
NORM <Master Drive Position in Counts> CAM
```

E) Restore the position units to their original values.

For example, if you knew the follower to be 3.00 inches and you knew the drive to be at 90 degrees:

A) Drive position = 90 degrees.

B) Drive position = 8192 counts.

C)
```
PNUM=1
PDEN=1
```

D)
```
EN
NORM 8192 CAM
```

E) Restore PNUM and PDEN

Finally, you must turn GEAR on. This connects the drive to the follower. If you want tot test your system before you connect the master-drive, you can use VOFF. VOFF is the offset speed for the electronic gearbox. For example, if the master drive is not moving and you turn GEAR ON and set VOFF to 100 RPM, the position command will increase at a rate of 100 RPM. This has the same effect as the encoder option input running at 100 RPM. For a 12-bit system, this is equivalent tot the master drive rotating at 409,600 counts per minute; as 360 degrees is 32.768 counts, this is equivalent to 12.5 (409,600/32,768) cycles through the cam table every minute.

PCAM and PCMD

The BDS5 uses a special variable for camming, PCAM. PCAM is the position command from the cam table. This is usually the role for PCMD (position command). However, when camming is enabled, PCM represents the position from the electronic gearbox; that is, the position that goes into the table. PC M is the output from the table.

PCMD is automatically in a "ROTARY" mode where the distance of one rotation is fixed at 32.768 counts. PCAM can be printed or recorded with PC-Scope. In fact, if you want to see your cam profile, you can record PCAM and PCMD simultaneously. For example, the following line records both positions for 0.5 seconds.

```
RECORD  500  1  PCMD PCAM
```

You can then use PC-Scope to verify your profile. Also, the PS and RS commands display "CAMMING" if GEAR and CAM are both ON.

Limitations

There are several limitations for camming applications. These limitations reflect that many functions of the BDS5 are not useful when caming. For example, profile commands (MI, MA, J, JT, JF, MCGO) are not allowed. ROTARY must be OFF. Any error that disables the drive also disables camming; you must re-normalize after such errors. Error 23, SOFTWARE OVERTRAVEL, disables the BDS5 and breaks the user program; when camming is off, this error only breaks the user program.

CAMMING DETAILS

by George Ellis 7/92
(Edited by Rick Furr June 02, 1993)

Camming is implemented as a modification of the BDS5 gearbox. As Figure 5 shows, the standard gearbox produces PCMD by multiplying PEXT by the ratio GEARI/GEARO. PE is formed by subtracting PFB from PCMD. Usually, PEXT is generated from another motor's feedback sensor. In this way, a master motor position (PEXT) controls the slave motor position (PFB).



Figure 5. BDS5 Gearbox

This method of controlling motors is limited. The only profile that is allowed is one where the slave position is proportional to the master position. Often, applications require that a master motor will be turning at a relatively constant speed, but the slave motor must execute a profile. Actually, the BDS5 has "Profile Regulation", a mode where the rate of the slave profile is tied to the master speed. At first we thought this would work for camming. Unfortunately, each time a new profile is started, there are a few milliseconds when the master position is ignored. So, although the rate of the profile is controlled by the master, the master phase and slave phase are not locked together. Over time, the master position drifts with respect to the slave position.

To implement camming, a new approach had to be taken. We decided to modify the gearbox by adding a look-up table. As Figure 6 shows, PEXT is processed by the gearbox to form PCMD. PCMD is then used as an index into a CAM look-up table to produce a new variable, PCAM. When camming is enabled, PCAM is used to form PE.

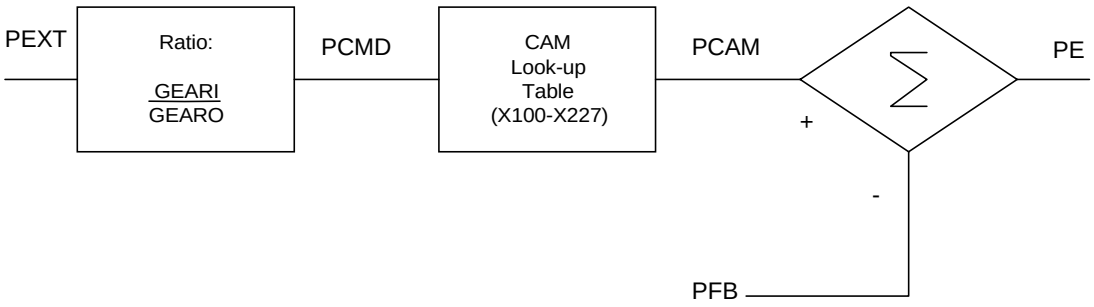


Figure 6. BDS5 Camming

Interpolation

The BDS5 CAM Table only has 128 points but with the help of interpolation, the BDS5 is able to generate a 32,768 point CAM. The interpolation algorithm will split each CAM table point into 256 linearly interpolated mid-point positions based on the master input. this is why the gear ratio must be chosen so that each revolution of the Master Cam input generates 32.768 counts of PCMD to the CAM table. This scaling is easily accomplished by programming the GEARI variable to 32,768 and then programming the number of counts generated by one revolution of the master input CAM into GEARO. A example would be as follows:

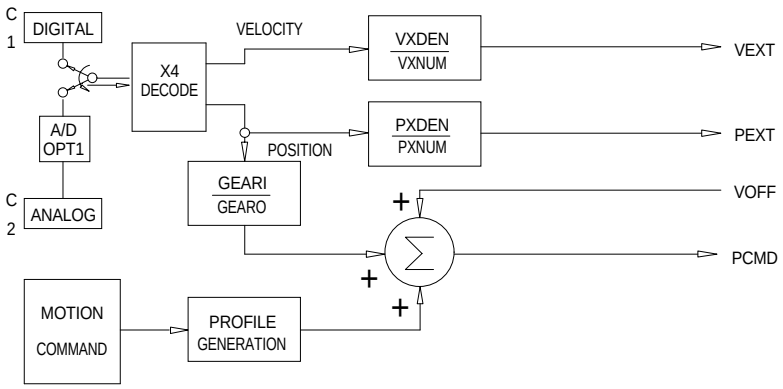
The master input device is a 2,000 line encoder. Each revolution of this encoder would produce 8,000 (2,000x4) counts of position command. If it took two revolutions of the mater to make one turn of the CAM then the master input would receive 16,000 (8,000x2) counts of position command per turn of the CAM. this means that the variable GEARO should be programmed to 16,000

Note: The variable GEARO must be number between 0 and 32,767. The variable GEARI must be a number between -32,768 and 32,767.

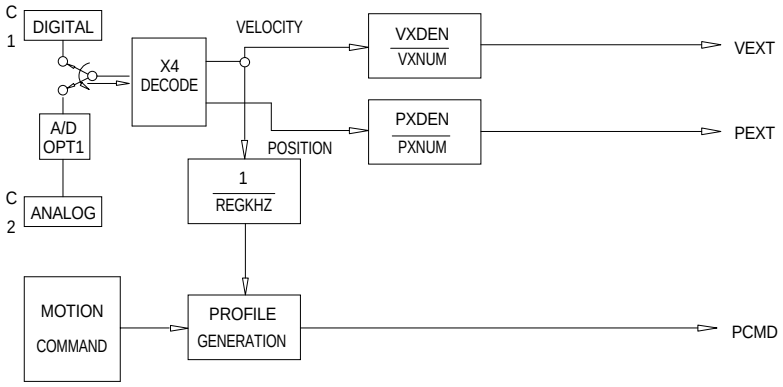
BDS5 MASTER/SLAVE

The next page will provide a more detailed block diagram of the Electronic Gearbox, Profile Regulation, and Electronic CAM master/slave modes of the BDS5.

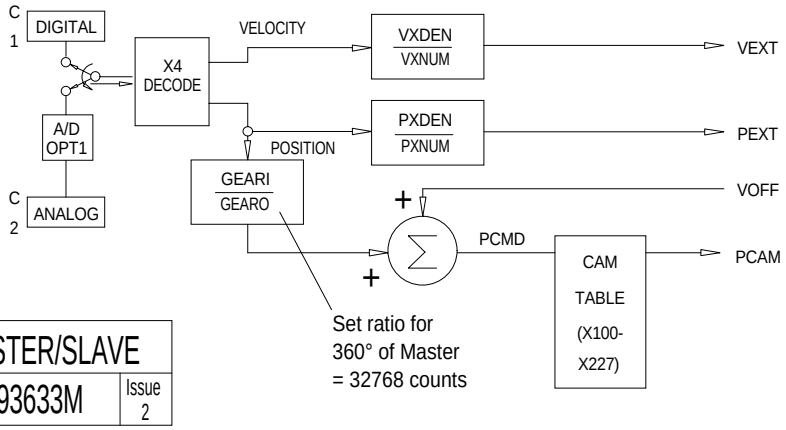
ELECTRONIC GEARBOX



PROFILE REGULATION



ELECTRONIC CAM



BDS5 MASTER/SLAVE		
RICK FURR	A-93633M	Issue
06-03-93		2

TESTING

A simple test program was written. This test performs the following task:

1. Loads X100-X227 with a triangle wave where x100 = 0, x101 = 100, x102 =200, ... The mid point of the triangle wave is X164 so that X163 = 6300, X164 = 6400, X165, 6300, X166 = 6200, ... [Lines 14-24]

2. Enable the BDS5 and enable camming [Lines 28-32].

3. Use VOFF to move PCMD through the cam cycle. VOFF is usually used with the gearbox to add an offset speed. It was designed for use with analog input where a customer may need to add an offset speed to adjust out error from a D/A converter. Here, we use it to simplify the test (without VOFF, testing would require a second motor be connected to the gearbox).

4) Loop control [Lines 35-37, 58-61]

5) Calculate which segment this iteration is in (it repeats from segment 0 to segment 127 every 128 iterations) and store it in X2. [Line 38]

6) Determine PCMD at the end of the segment. [Line 40]

7) Wait until PCMD reaches the boundary. Store the commanded position (PCAM) [Lines 45-53]

8) Calculate error between what the command is (X4) and what is should be (X(X2)).

9) Subroutines to print whether iteration tested good or bad [Lines 63-70].

CAM TEST PROGRAM FOR THE BDS5

```
1   ;CAM TEST       ghe 7/16/92
2   ;
3   ;use voff set to keep vcmd moving at a constant speed.  Selected that
4   ;speed to be 20 RPM which is about 1365 counts/second.  The entire
5   'cam cycle is 32768 counts.  Each of the 128 segments is 256 counts.
6   ;So. 20 RPM cycles through the cam cycle at one cycle per 32768/1365
7   ;seconds or 24 seconds.  Each of the 128 cycles requires 24/128 or
8   '187 msecs.
9
10  ;1$
11  ;This section loads cam variables with a triangular profile where
12  ;each segment is different from the last by 100 counts
13  ;First, load variables x100-x164
14  ;xi = 100                        ;starting variable
15  ;2$                          ;loop beginning
16  X(X1) = = (X1-100)*100  ;load the "up side" of the triangle
17  X1 = X1+1                   ;increment the loop counter
18  ? x1 1e 164 goto 2          'test loop--keep going until X164
19  ;
20  ;Now, load variables X165-X227 with the "down side" of the counter
21  3$                          ;loop beginning
22  X(X1) = X164-(X1-164)*100
23  X1 = X1+1                   ;increment the loop counter
24  ? X1 LE 227 GOTO 3;test loop--keep going until X227
25  ;
26  ;Now, it's time to start the CAM
27  4$
28  PLIM OFF                    ;Standard line to enable
29  GEAR OFF                    ;Disable GEAR so we can enable CAM
30  EN
31  NORM 0 CAM                  ;Normalize and enable CAM
32  GEAR ON                     ;Now, we can enable GEAR
33  VOFF 20                     ;Use offset speed of 20 RPM to go through
34                              ; cam cycle
35  B                           ;Break back to Immediate mode
```

BDS5 SERIAL COMMUNICATIONS 1

DOC=BDS5COM1.DOC Rev 3 June 03, 1993

The following chart shows the various ASCII codes and formats that will be used in all examples in this document.

CONTROL CODE DEFINITIONS				
Name	**Symbol**	**Control**	**Hex**	**Decimal**
Bell	<bell>	^F	07h	7
Backspace	<bs>	^G	08h	8
Line Feed	<lf>	^J	0Ah	10
Carriage Return or Enter	<cr>	^M	0Dh	13
Escape	<esc>	^[	1Bh	27
Space	<sp>	--	20h	32
xx - Number of Spaces	<xx-sp>	--	20h	32
Comment Statements	{Comments}	---	---	---
Repeating or Additional Data	{.....}	---	---	---

BDS5 EXAMPLE FORMATS

All examples will have the following format:

<Initial Prompt> <Host Command>
<BDS5 Echo>
<Response Data>
<...>Final Prompt>

The BDS5 is designed to easily interface with a human (via a standard dumb terminal) and is also designed to interface to a general purpose computer. Part of this design involved the selection of unique prompts for each of the BDS5's 9 modes. These unique prompts will allow the computer to determine the BDS5's current mode.

BDS5 NON-MULTIDROP PROMPTS	
PROMPT	**DEFINITION**
-->	INTERACTIVE MODE
==>	MONITOR MODE
s->	SINGLE STEP MODE
t..	TRACE MODE
l->	PROGRAM UPLOAD
e->	INTERNAL EDITOR
i->	EDITOR INPUT
f->	EDITOR FIND
c->	EDITOR CHANGE

Every prompt is preceded by a <cr><lf>.

The BDS5 will print a prompt to the serial port whenever it is ready to receive a character. The BDS5 will echo each character transmitted to it.

The BDS5 will never print data to the serial port while a prompt or input statement is active. This means that if an error occurred the BDS5 would hold the error message if a prompt was present or if an input statement was active.

All error messages respond with the first three characters of "ERR".

The BDS5 can be programmed in the user program to print out data with the "PRINT" statement and the background (BACKGROUND$) routine can be programmed to print out data on an asynchronous interval with the "PRINT" statement.

The BDS5 can be programmed to enter an autobaud sequence on power-up or can be programmed to a fixed baud rate on power-up. By setting "ABAUD=1" the BDS5 will autobaud on power-up. After autobauding, the variable "BAUD" will contain the current baud rate.

The BDS5 can also be set to automatically power-up at a fixed baud rate. By setting "ABAUD=0" and "BAUD=9600" the BDSD5 would power-up at 9600 baud.

The BDS5 can be reset to it's default serial conditions (Autobaud) by holding the MOTION input off during power-up. This allows communications to be reestablished it au incompatible baud rate was programmed into the BDS5.

RS-232 COMMUNICATIONS -- INTERACTIVE MODE

The interactive mode is the normal state the BDS5 will be in when it is not running a program. The following are typical BDS5 response sequences.

The BDS5 will respond to a <cr> with a "<cr><lf>-->" character string while in the interactive mode. The interactive prompt (-->) indicated that the BDS5 is ready for the next command.

```
-->"<cr>"          "<cr><lf>"
                   "-->"
```

The BDS5 generates the same response with the <esc> character as it does with the <cr> (see above) while in the interactive mode.

```
-->"<esc>"         "<cr><lf>"
                   "-->"
```

The Jog command or any typical BDS5 command will generate the following response:

```
-->"J 1000<cr>"    "J 1000<cr><lf>"
                   "-->"
```

The Print command can be used to print the contents of any BDS5 variable. Remember that variables print right justified so up to 11 Spaces could precede the value to be printed.

```
-->"P SEG<cr>"     "P SEG<cr><lf>"
                   "<11-sp>0<cr><lf>"
                   "-->"
```

The BDS5 Print command can specify a format to use for printing the variable. The only restriction is that if the format is too small "X"s will be printed.

```
-->"P SEG[1]<cr>"  "P SEG[1]<cr><lf>"
                   "0<cr><lf>"
                   "-->"

{ASSUME PFB=1024}
-->"P PFB<cr>"     "P PFB<cr><lf>"
                   "<8-SP>1024<cr><lf>"
                   "-->"
```

-->"P PFB[3]<cr>" "P PFB[3]<cr><lf>"
"XXX<cr><lf>"
"-->"

-->"P PFB[4]<cr>" "P PFB[4]<cr><lf>"
"1024<cr><lf>"
"-->"

Any programmable variable (user and dedicated can be programmed with a value by using the BDS5 equate function.

X1 1000" "X1=1000<cr><lf>"
"-->"

The BDS5 equate function will work with either an Equal Sign (=) or a Space.

X1 1000" "X1 1000<cr><lf>"
"-->"

The Run command can be used to enable the BDS5 multitasking by typing "RUN". Notice that once the BDS5 has started running that the prompt does not return.

-->"RUN<cr>" "RUN<cr><lf>"

The RUN command can also start a specific program label.

-->"RUN<cr>" "RUN<cr><lf>"

If an error occurs it will not print out if a prompt is present until a <cr> is received.

-->"<cr>" "<cr><lf>"
"<bell>ERR 17 FEEDBACK LOSS<cr><lf>"
"-->"

If the host was sending a command to the BDS5 and an error occurred before the command was finished the BDS5 will ignore the command and respond with the following sequence.

```
-->"<cr>"               "<cr><lf>"
                        "-->"

{NOW AN ERROR OCCURS}

{TYPE IN A COMMAND AND A <cr>}
-->"P PFB<cr>"          "<cr><lf>"
                        "ERROR MSG WAITING--COMMAND IGNORED<cr><lf>"
                        "<cr><lf>"
                        "<lf>"
                        "<bell>ERR 17 FEEDBACK LOS<cr><lf>"
                        "-->"
```

After an error with a severity level high enough to disable the drive the fault light will turn on. to clear this light simply re-enable the BDS5 with the Enable command. The Enable command takes a few seconds to complete and returns a prompt.

```
-->"EN"                 "EN<cr><lf>
                        <...time delay...>
                        "-->"
```

RS-232 COMMUNICATIONS -- RUNNING A PROGRAM

If a program is running -- then an ESCAPE character must be sent to get the attention of the BDS5. The monitor mode will be active on getting the attention of the BDS5. A second ESCAPE will drop the BDS5's attention and exit the monitor mode. The monitor mode will accept a sub-set of the BDS5 commands. This subset includes:

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

To enter the monitor mode while running a program send an <esc> character.

"<esc>" "<cr><lf>"
"ENTER MONITOR MODE. PUSH ESCAPE TO EXIT <cr><lf>"
"==>"

If the response is as follows then the <esc> has caused the BDS5 to exit the monitor mode. If this is the case then send another <esc> to reenter the monitor mode.

"<esc>" "<cr><lf>"
"ENTER MONITOR MODE.

A single <cr> will return the following response. If there is no response then the monitor mode is not active and an <esc> should be sent tot he BDS5 to enter the monitor mode.

"<cr>" "<cr><lf>"
"==>"

A Stop command will stop motion at AMAX and break the program. The interactive prompt will be returned indicating that the BDS5 has stopped running the program.

==>"S" "S<cr><lf>"
"-->"

A Break command will stop at AMAX and break the program. The interactive prompt will be returned indicating that the BDS5 has stopped running the program.

==>"B" "B<cr><lf>"
"-->"

A Kill command will disable the motor and break the program. The interactive prompt will be returned indicating that the BDS5 has stopped running the program.

```
==>"K"                  "K<cr><lf>"
                        "-->"
```

Sending a command not allowed in the monitor mode will return the following response.

```
==>"J 1000 <cr>         "J 1000<cr><lf>"
                        "<cr><lf>"
                        "<lf>"
                        "<bell>ERR 63 'J 1000'<22-sp>NOT AT THIS LEVEL<cr><lf>"
                        "==>"

==>"P SEG[1]"<cr>       "P SEG[1]<cr><lf>"
                        "0<cr><lf>"
                        "==>"

{ASSUME PFB=1024}
==>"P PFB<cr>"          "P PFB<cr><lf>"
                        "<8-sp>1024<cr><lf>"
                        "==>"

==>"P PFB[4]<cr>"       "P PFB[4]<cr><lf>"
                        "1024<cr><lf>"
                        "==>"

==>"P PFB[5]"           "P PFB[5]<cr><lf>"
                        "<sp>1024<cr><lf>"
                        "==>"

==>"P PFB[3]"           "P PFB[3]<cr><lf>"
                        "XXX<cr><lf>"
                        "==>"

{THE XXX MEANS THAT PFB WAS TOO BIG TO FIT INTO 3 SPACES}

==>"X1=1000"            "X1=1000<cr><lf>"
                        ==>

{THIS SETS VARIABLE X1 TO 1000}

==>"X1.1000"            "X1.1000<cr><lf>"
                        ==>

{THIS SETS VARIABLE X1 TO 1000}
```

==>"RUN<cr>" "RUN<cr><lf>"
"<cr><lf>"
"<lf>"
"<bell>ERR 63 'RUN'<28-sp>NOT AT THIS LEVEL<cr><lf>"
"==>"
{THE RUN COMMAND IS NOT ALLOWED IN THE MONITOR MODE}

A ^X will act like a "S"top and a "B"reak command. The BDS5 will respond to this command while in the monitor mode, while in the interactive mode, or while running program.

==>"RUN 6" "RUN 6<cr><lf>"
{NOW A PROGRAM IS RUNNING AND NO PROMPT IS PRESENT}

{NOW AN ERROR OCCURS}
"<cr><lf>"
"<lf>"
"<bell>ERR 17 FEEDBACK LOSS<cr><lf>"
"-->"

==>"RUN 6" "RUN 6<cr><lf>"
{NOW A PROGRAM IS RUNNING AND NO PROMPT IS PRESENT}

{TYPE AN ESCAPE TO ENTER MONITOR MODE}
"<esc>" "<cr><lf>"
"<lf>"
"ENTER MONITOR MODE. PUSH ESCAPE TO EXIT<cr><lf>"
"==>"

{NOW AN ERROR OCCURS BUT A PROMPT IS PRESENT SO}

{ THE ERROR WILL NOT PRINT OUT}
{TYPE A <cr>}
==>"cr>" "<cr><lf>"
"<lf>"
"<bell>ERR 17 FEEDBACK LOSS<cr><lf>"
"-->"

```
==>"RUN 6"              "RUN 6<cr><lf>"
{NOW A PROGRAM IS RUNNING AND NO PROMPT IS PRESENT}

{TYPE AN ESCAPE TO ENTER MONITOR MODE}
"<esc>"                 "<cr><lf>"
                        "<lf>"
                        "ENTER MONITOR MODE.  PUSH ESCAPE TO EXIT<cr><lf>"
                        "==>"

{NOW AN ERROR OCCURS BUT A PROMPT IS PRESENT}
{SO THE ERROR WILL NOT PRINT OUT}

{TYPE IN A COMMAND AND A <cr>}
==>"P PFB<cr>"          "<cr><lf>"
                        "ERROR MSG WAITING--COMMAND IGNORED<cr><lf>"
                        "<cr><lf>"
                        "<lf>"
                        "<bell>ERR 17 FEEDBACK LOSS<cr><lf>"
                        "-->"
```

UPLOADING & DOWNLOADING PROGRAMS

The BDS5 has two commands to allow the up-loading and down-loading of the BDS5 program memory.

"<BDS" will print to the serial port each line of the program memory. Each program line is terminated with a <cr><lf>. This command terminates with a "<cr><lf>-->" prompt.

```
-->"<BDS<cr>"          "<BDS<cr><lf>"
                       "<Program Line #1><cr><lf>
                       "....."
                       "<Last Program Line #><cr><lf>"
                       "-->"
```

">BDS" will place the BDS5 into the program upload mode. On getting the first <cr> terminated line the program memory will be erased. Each <cr> terminated line will respond with a "<cr><lf>|->" prompt. Sending an <esc> tot he BDS5 will end the program upload mode. Upon completing a program upload the host may request a special BDS5 checksum of the data stored in memory. After receiving this special checksum the host may in future uploads use this value to verify the BDS5 program has been uploaded error free.

```
-->">BDS<cr>"          "BDS<cr><lf>"
                       "l->"
{THE l-> PROMPT INDICATES THE BDS5 IS READY TO RECEIVE A PROGRAM}

|->"<your first program line><cr>" "<your first program line><cr><lf>"
|->"<.....>"                      "<.....>"
|->"<your last program line><cr>"  "<your last program line><cr><lf>"
|->"<esc>"                        "<cr><lf>"
                                  "-->"
```

Recalculation of this checksum by the host is not possible because this is a special word checksum of the BDS5's internal program memory with a varying offset. As stated earlier the most efficient method of exploiting this checksum is to upload the BDS5 program, print out the checksum variable, and then use that value for comparison of future uploads.

```
-->"P CHECKSUM<cr>"  "P CHECKSUM<cr><lf>"
                     "<7-sp>65280<cr><lf>"
                     "-->"
{YOU CAN ALSO PRINT THE CHECKSUM IN HEX}

-->"P CHECKSUM[H]<cr>"  "P CHECKSUM[H]<cr><lf>"
                        "<4-sp>FF00H<cr><lf>"
                        "-->"
```

UPLOADING & DOWNLOADING SYSTEM VARIABLES

The BDS5 can print to the serial port two groups of internal variables. The first group is all of the variables. The second group is a subset consisting of compensation specific variables.

"DUMP" will print to the serial port all variables from the BDS5.

"DUMP TL" will print to the serial port the compensation specific variables from the BDS5.

To upload non-factory variables to the BDS5 simply transmit each variable followed by a Space or Equal Sign followed by the numerical value and terminated by a <cr>. Do not transmit the next variable line until a prompt has been received. The prompt indicates that the BDS5 is ready to receive the next variable or command.

-->"BAUD 9600<cr>" "BAUD 9600<cr><lf>"
"-->"

-->"ABAUD=1<cr>" "ABAUD=1<cr><lf>"
"-->"

-->"X1=111234<cr>" "X1=111234<cr><lf>"
"-->"

Not all BDS5 variables are programmable and some variables are programmable at the factory only. The factory protected variables contain motor specific information.

MULTIDROP COMMUNICATIONS

Multidrop communications will allow you to have up to 32 axes on serial communications line. Each axis must have an unique address. The valid addresses are "0"-"9" and "A"-"Z". The Axis Attention Character is a Backslash "\". The Backslash when followed by a valid Axis ID will wake up the addressed axis. "*" is a special Axis Broadcast All command that will wake-up all the axes to respond to any transmitter command but no axis may transmit. The BDS5 prompts are slightly different from the standard RS-232 prompts. The Multidrop prompts always print the Axis ID as their first character. The normal first character is then slid over to the second position in the prompt. The prompt terminates with a ">".

BDS5 MULTIDROP PROMPTS	
ASSUME: ADDR=65 AXIS I.D.="A"	
PROMPT	**DEFINITION**
A->	INTERACTIVE MODE
A=>	MONITOR MODE
As>	SINGLE STEP MODE
At.	TRACE MODE
Al>	PROGRAM UPLOAD
Ae>	INTERNAL EDITOR
Ai>	EDITOR INPUT
Af>	EDITOR FIND
Ac>	EDITOR CHANGE

All handshaking will be identical to the above examples except the prompts will change as shown in the above chart.

{WAKE UP AN AXIS "A" (ADDR=65)}

"\A<cr>" "<cr><lf>"
"A->"

BDS5 SERIAL COMMUNICATIONS 2

DOC=BDS5COM2.DOC Rev 4 July 13, 1993

- The serial protocol is simple ASCII with full duplex echo.
  1. The simple ASCII protocol was chosen to allow easy communications with any ASCII device. I.e. dump terminals, hand held terminals, panel mounted terminals, IBM compatibles running terminal emulation software, etc.
  2. With full duplex echo, the BDS5 will transmit each character received back to the host device. This allows the host to verify each character was correctly sent.

- The BDS5 will transmit a unique prompt when it is ready to receive commands. The interactive prompt will be "-->". If a prompt is not present, then the BDS5 is not listening to the serial port.

BDS5 PROMPTS		
MODE	**NON-MULTIDROP (ADDR=0)**	**MULTIDROP (ADDR=65)**
INTERACTIVE	-->	A->
MONITOR	==>	A=>
SINGLE-STEP	s->	As>
TRACE	t..	At.
EDIT	e->	Ae>
LOAD	1->	A1>
EDIT/INSERT	i->	Ai>
EDIT/FIND	f->	Af>
EDIT/CHANGE	c->	Ac>

- The following control codes will be used throughout this discussion:

CONTROL CODE DEFINITIONS				
NAME	**SYMBOL**	**CONTROL**	**HEX**	**DECIMAL**
Acknowledge	<ack>	^F	06h	6
Bell	<bell>	^G	07H	7
Backspace	<bs>	^H	08h	8
Linefeed	<lf>	^J	0Ah	10
Carriage Return or Enter	<cr>	^M	0Dh	13
Not Acknowledge	<nak>	^U	15h	21
Escape	<esc>	^[	1Bh	27
Space	<sp>	--	20h	32

Some of the more important serial commands that have been available in all versions of software are in the following table:

BDS5 SERIAL COMMANDS **(Firmware version 2.0 and later)**	
PROMPT 0|1	Used to suppress the printing of the three character prompt to the serial port. PROMPT is remembered on Power-up.
<variable>=<expr>	Used to equate any system variable to any valid math expression. Valid math expressions include user variables, indirect references to user variables, constants, algebraic and logical math operators, parentheses. Parentheses can be nested up to two levels deep. Spaces are not allowed in expressions.
P <expr>{[<format>]} | "<text>" {...}	Print command - Used to print strings and variables specified with the optional format terminated by a <cr><lf> sequence. Also see the "R", "RS", "PA", and "PAS" print commands.
PS <expr>{[<format>]} | "<text>" {...}	Print Status command - Used to print strings and variables specified with the optional format terminated by the drive status and a <cr><lf> sequence.
R <expr>{[<format>]} | "<text>" {...}	Refresh command - Used to print strings and variables specified with the optional format terminated by a <cr>. This command identical to the P command except there is only a terminating <cr>. Also see the "R", "PS", "PA", and "PAS" print commands.
RS <expr>{[<format>]} | "<text>" {...}	Refresh Status - Used to print strings and variables specified with the optional format terminated by the drive status and a <cr>.

BDS5 COMMUNICATIONS

The BDS5 will communicate over an RS-232 serial bus with simple ASCII commands.

- Each ASCII character transmitted to the BDS5 will be echoed back to the host. This echo will allow a simple character by character verification that the data was properly received.

- Each ASCII command string will be terminated by an ASCII carriage return (<cr>).

- Upon receiving the <cr> the BDS5 will acknowledge the command by transmitting a "<cr><lf>" followed by a system prompt ("-->"). The prompt is used to indicate what mode is currently active in the BDS5 and that the BDS5 is ready to receive another command.

SERIAL COMMAND EXAMPLES:

“ILIM=50<cr>”	This would set the current limit to 50%.
”OUT=1FH<cr>”	This would set the 8 user programmable outputs to 1F hex.
”EN<cr>”	This would enable the BDS5.
”DIS<cr>”	This would disable the BDS5.
”J 1000<cr>”	This would Jog an enabled axis at 1000 using the acceleration and deceleration rates programmed into ACC and DEC.
”P PFB<cr>”	This would print the feedback position to the serial port.
”P PFB[4]<cr>”	This would print the feedback position formatted to 4 places to the serial port, terminated by a <cr><lf>.
”R PFB[H]<cr>”	This would print the feedback position in hex format to the serial port terminated by a <cr>.
”R PFB[H5}<cr>”	This would print the feedback position in 5 places using hex format to the serial port terminated by a <cr>.
”PA” PFB<cr>”	This would print the feedback position to the serial port terminated by no characters.

Additionally, for more communications capabilities, the following commands have been added to firmware version 3.0.0:

NEW BDS5 SERIAL COMMANDS **(Firmware Version 3.0.0 and Later)**	
MONITOR 0|1	This will automatically force entry into the Monitor mode at the start of running a program. MONITOR is set to 0 on Power-up.
ECHO 0|1	Used to suppress the echo of serial port characters. ECHO is remembered on Power-up.
MSG 0|1	Used to suppress the serial Power-up message and the Monitor mode message. MSG is remembered on Power-up.
PA <expr>{[<format>]} | "<text>" {...}	Print Append command - Used to print strings and variables specified with the optional format. This command identical to the P command except there is no terminating <cr><lf> sequence. Also see the "P", "PS", "R", and "RS" print commands.
PAS <expr>{[<format>]} | "<text>" {...}	Print Append Status command - Used to print strings and variables specified with the optional format terminated by the drive status. This command is identical to the PS command except there is no terminating <cr<lf> sequence.

Note, the BDS5 now has a full set of Print commands. These commands are P, PS, R, RS, PA, and PAS. The only differences being the P and PS commands terminate the output string with a <cr><lf>, the R and RS commands terminate the output string with just a <cr>, and the PA and PAS commands do not terminate the output string at all.

The Monitor mode switch will cause the BDS5 to automatically enter the Monitor mode upon the execution of a BDS5 program. This will in essence cause the BDS5 to always have the serial port active and waiting for a command. Previously BDS5 required an <esc> character to be received in order to activate the serial port while a user program was running.

APPENDIX A -- SERIAL PORT

STANDARD: RS-232 & RS-485 SERIAL
CONNECTOR: C1 - 9 PIN DB-9
SIGNALS (4): SHIELD, RECEIVE, TRANSMIT, COMMON
MODES: 8 BIT ASCII, NO PARITY

BDS5 SERIAL CONNECTOR	
PIN	**NAME**
1	SHIELD
2	REC
3	XMIT
4	N/C
5	COMMON
6	TD+
7	TD-
8	RD+
9	RD-

- The BDS5 has one RS-232 serial DB-9 male connector.
- The communications format will be 8-bit ASCII with 1 start bit, 1 stop bit and no parity bit.
- The communications data rate will be from 300 to 19,200 baud (bits-per-second). This rate is programmed in the variable "BAUD".
- This BDS5 can autobaud to determine the communication rate when the flag "AUTOBAUD" is enabled. Autobauding is performed by the BDS5 receiving a series of <cr>.
- The serial connector is optionally configurable to RS-485. RS-485 is enabled with the addition of RS-485 receiver/driver chips and setting a valid multi-drop address in the variable "ADDR".

BDS5 MULTI-DROP PROTOCOL	
\<address>	Multidrop Axis Address Valid addresses are 0-9 and A-Z.
\\	Multidrop Hang-up Axis
*	Multidrop Broadcast All

FIRMWARE UPGRADE NOTICE 3.02b

FEBRUARY 02, 1994

The Engineering department is releasing new beta level proms that fix some operational problems with error recovery. These proms are replacements for the current factory default prom version 3.0.1. The new prom is labeled 3.02b. The "b" indicates this prom is still in beta release until the BDS5 firmware qualification tests are finished. This prom has passed our initial qualification tests. This prom is being released early in an attempt to help our customers. Please immediately forward any feedback from testing this prom to our field service group, your success is very important to us.

The following changes have been made to the firmware:

- The user flag "MSG", when cut off, will suppress any printing of error messages. (This is in addition to its suppressing the power-up message, the running program message, and the Entering and Exiting Monitor mode message.)
- The Monitor mode will now be properly canceled by any error that is not directly caused by the Monitor mode. This fixes the problem with the error handler hanging until control of the serial port is returned from the Monitor mode, thus allowing the error message to print to the serial port. Now the printing will be allowed and the error handler, as a final step can pass execution to the ERROR$ label. This modification also prevents a print command from within the ERROR$ from hanging the error handler.

NOTE: Since an error will cancel the Monitor mode, multiple errors in fast succession could cause problems with entering the Monitor mode in order to stop the user program with a Break ("B") or Kill ("K") command. The method to stop program execution in this case is to issue a "^X" (18h) command. This should break the user program reliably.

- The error handler has been enhanced to allow execution of ERROR$ to completion even in the event of additional errors. Before this the BDS5 was designed to automatically break program execution on the second error.
- We also found that a Software watchdog error could be triggered if an error occurred during the execution of the firmware code that sets up a Jog command. This has been fixed.

Additional Note: The Beta version of the firmware is dedicating user register X100 and X101 to capture diagnostic information during any error. Please avoid using this register if possible.

Error Handler (ERROR$)

When a serious error occurs, the BDS5 breaks execution of your program and checks your program to see if you entered ERROR$. If you did, the error handler (that is, the routine that follows the ERROR$) is executed. All multi-tasking is suspended, including alarms, when the error handler is being executed. See Chapter 10 for more information on the user's error handler.

Errors can also cause the BDS5 to change modes. Some errors are serious enough to cause the BDS5 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. ERROR$ is provided as a graceful end to program execution and not as a means of automatically restarting the program. The program can be restarted using the AUTO$ and the CYCLE input. For example, you can set outputs or print a message. It is not intended to continue the program as if the error never occurred.

ERRORS

The BDS5 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.

The BDS5's response to an error depends on the error's severity. There are four levels of severity, listed below in increasing order:

ERROR SEVERITY LEVELS AND ACTIONS	
1.	Errors which cause warnings.
2.	Errors which cause a program break and stop motion, in addition to Level 1 Actions.
3.	Errors which disable the system and set the FAULT LED, in addition to Level 2 Actions.
4.	Errors which disable almost all BDS5 functions (including communications) and flash the CPU LED to indicate the error number. These are called firmware errors.

ERROR MESSAGES

When any error except a firmware error occurs, a message is displayed to 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 BDS5 will respond with:

ERR 50   'J 100'   BDS5 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 BDS5 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.

DEP01

If your BDS5 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 BDS5 provides the DEP switch, which, when turned on, cuts all error messages down to 40 characters. If your BDS5 prints to a DEP-01, type "DEP ON"

ERROR HISTORY

The BDS5 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 BDS5 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.

DISPLAYING ERROR MESSAGES

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

ERR 50

The BDS5 responds with:

ERR 50 BDS5 INHIBITED

You may display messages for errors from 1 through 999. If you type in an error number that the BDS5 does not recognize, it will respond with:

ERROR NOT FOUND.

A description of all errors is given in the BDS5 manual in Appendix D.

THE USER'S ERROR HANDLER

When an error occurs, the BDS5 decides what needs to be done: disable the drive, print out a message, and so on. This is called an error handler. Often, an application may require that other actions be taken. A common example is setting the Output (O) word to turn off some auxiliary machine such as a pump. These actions, which are very specific to the application, must be handled in the user's program. In effect, you can write an error handler for your application.

The User's Error Handler begins at the special label, ERROR$. Any error that breaks execution of your main program will restart execution from ERROR$, if it exists. The error handler provides instructions that need to be executed if an error interrupts your program. Examples of error handler instructions include setting outputs, printing messages, and storing information in user variables.

There are several restrictions that apply to the error handler; for example, GOTO, GOSUB, and RET are not allowed. Normally, ERROR$ should be at the end of your program. ERROR$ cannot be followed by POWER-UP$, AUTO$, MANUAL$, or any general purpose labels (0$ through 500$).

Software travel limits are automatically enabled on power-up. If you use these limits, you may have short sections of your program that disable them briefly. If an error occurs in one of those sections, software limits will remain disabled until you enable them or until the next power-up. For this reason, you should always enable travel limits from the User's Error Handler if they are disabled from any section of your program.

WARNING

BE SURE TO TURN PLIM ON FOR THE FOLLOWING CONDITIONS!

USE ERROR$ TO TURN PLIM ON FOR THE FOLLOWING CONDITIONS:

1) your application relies on Software Travel Limits.

2) your program disables Software Travel Limits at any point, even briefly.

ERROR$ is provided as a graceful end to program execution and not as a means of automatically restarting the program. The program can be restarted using the AUTO$ and the CYCLE input.

Use the Editor to enter the following example program:

```
12$
JUNK                ;this line generates an error
B
ERROR$
P  "RUNNING SIMPLE ERROR HANDLER"
OUT  0              ;turn outputs off
B
```

Now exit the Editor and type:

RUN 12

and the response should be:

ERR 80 LINE 2 'JUNK' INVALID COMMAND
RUNNING SIMPLE ERROR HANDLER

LSTLBL

The BDS5 stores the last label that is executed in the variable LSTLBL. You may need to know the last label that was executed to determine where the program terminated. Labels are stored in LSTLBL as follows:

Last label executed	Value Stored in LSTLBL
0-500	0-500
A$	501
B$	502
C$	503
VARIABLE$	504
BACKGROUND$	505
CYCLE$	506
ERROR$	not stored
MANUAL$	508
POWER-UP$	509

LSTERR

The variable LSTERR contains the error number of the most recent error. LSTERR is especially useful in the User's Error Handler (the lines that follow ERROR$). It allows your error handler to take special action based on a particular error number. For example, the error handler from above can be expanded to include LSTERR:

```
12$
JUNK                ;this line generates an error
B
ERROR$
P  "SIMPLE ERROR HANDLER"
OUT  0              ;turn outputs off
?  LSTERR  EQ  28  P  "THIS IS AN INVALID COMMAND"
B
```

Now exit the Editor and type:

RUN 12

and the response should be:

```
ERR  28   LINE 2   'JUNK'          INVALID COMMAND
SIMPLE ERROR HANDLER
THIS IS AN INVALID COMMAND
```

PFNL and Errors

If an error occurs that breaks your user program, you can use PFNL to tell you the last commanded position. This can be used to tell you where your program stopped. For example:

```
NORM  0
;if ERROR OCCURS here, PFNL = 0

MA  1000  100
;if ERROR OCCURS here, PFNL = 1000

MA  2000  1000
;if ERROR OCCURS here, PFNL = 2000, even if the first
;move is still in progress.
```

You can use PFNL to tell which moves were calculated before the error occurred. You can use PFB to tell which moves were actually processed.

FIRMWARE ERRORS

Firmware errors are an indication of a serious problem with the BDS5. 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 5. Contact the factory should one of these errors occur.

PROGRAM CYCLE AND ERROR RECOVERY USING AN ALARM WITH INPUT #1

```
1$                      ;same as power-up$
POWER-UP$               ;powerup auto start label
PLIM  OFF               ;turn off software limits
EN                      ;enable motor
END                     ;return to multi-tasking
;
A$  I1  ON              ;when input #1 = 1
MI  40960  100          ;move incremental
TIL  PFB  EQ  PCMD      ;wait for motion to stop

END                     ;return to multi-tasking
;
ERROR$                  ;error recovery
O1  ON                  ;turn O1 on
D  5000                 ;wait for 5 seconds to let
                        ;OK2EN settle
TIL OK2EN  EQ  1        ;wait for OK to Enable
O1  OFF                 ;turn O1 off
EN                      ;enable motor
END                     ;return to multi-tasking
```

PROGRAM CYCLE AND ERROR RECOVERY USING AUTO$ WITH THE CYCLE INPUT

```
1$                      ;same as power-up$
POWER-UP$               ;powerup auto start label
PLIM  OFF               ;turn off software limits
EN                      ;enable motor
END                     ;return to multi-tasking
;
AUTO$                   ;MANUAL=0, CYCLE=1
MI  40960  100          ;move incremental
TIL  PFB  EQ  PCMD      ;wait for motion to
stop
END                     ;return to multi-tasking
;
ERROR$                  ;error recovery
O1  ON                  ;turn O1 on
D  5000                 ;wait for 5 seconds to let
                        ;OK2EN settle
TIL OK2EN  EQ  1        ;wait for OK to Enable
O1  OFF                 ;turn O1 off
EN                      ;enable motor
END                     ;return to multi-tasking
```

PROGRAM CYCLE AND ERROR RECOVERY USING MANUAL$ WITH THE MANUAL INPUT

```
1$
POWER-UP$               ;powerup auto start label
PLIM  OFF               ;turn off software limits
EN                      ;enable motor
END                     ;return to multi-tasking
;
MANUAL$                 ;when MANUAL = ON
MI  40960  100          ;move incremental
TIL  PFB  EQ  PCMD      ;wait for motion to stop

END                     ;return to multi-tasking
;
ERROR$                  ;error recovery
O1  ON                  ;turn O1 on
D  5000                 ;wait for 5 seconds to let
                        ;OK2EN settle
TIL OK2EN  EQ  1        ;wait for OK to Enable
O1  OFF                 ;turn O1 off
EN                      ;enable motor
END                     ;return to multi-tasking
```

PROGRAM CYCLE AND ERROR RECOVERY USING RUN WITH INPUT #1

```
1$
POWER-UP$               ;powerup auto start label
PLIM  OFF               ;turn off software limits
EN                      ;enable motor
;
2$
? I1  EQ  OFF  GOTO  2  ;if I1=0 then loop
MI  40960  100          ;move incremental
TIL  PFB  EQ  PCMD      ;wait for motion to
stop
GOTO  2                 ;loop to label 2$
;
ERROR$                  ;error recovery
O1  ON                  ;turn O1 on
D  5000                 ;wait for 5 seconds to let
                        ;OK2EN settle
TIL OK2EN  EQ  1        ;wait for OK to Enable
O1  OFF                 ;turn O1 off
RUN  1                  ;force program restart at
                        ;label 1$
```

BDS5 FIRMWARE UPGRADE NOTICE

DOC=BDS5-304.DOC.R5 / FIRM-UP.DOC June 17, 1994

VERSION 3.0.4

UPGRADES

When upgrading older systems (pre version 3.0.0) with 3.0.4. Be sure to initialize the three new non-volatile flags for proper operation:

1. **ECHO=1**
2. **MSG=1**
3. **EXTDX=0**

CHANGES (from 3.0.3)

- Enhanced the CLEARX command to allow separate clearing of all User Registers or User Flags or both.

  Syntax: CLEARX {1|2}

  Example: CLEARX - Clear both User Registers and User Flags
  CLEARX 1 - Clear only the User Registers
  CLEARX 2 - Clear only the User Flags

- Enhanced the Break (B) command to allow separate the optional breakink of a user INPUT command:

  Syntax: B {I}

  Example: B - Break the user program
  B I - Break a user INPUT command (if active).

The "B I" command was designed to allow the user to break any active Input command because Printing from all tasks are suppressed until completion of the Input command. Aditionally, any Print while the Input command is active will HOLD the task at the Print command until the Input has finished. This means that if, for example, a user needed to Print a message to the machine operator during an Alarm condition, this would not work if an Input command was active. Now, with the "B I" command, the user can imbed the "B I" command in the program prior to any critical Print messages and thus cancel any active Input command.

Break Input Example	
1$	;Main loop
O1 1	Output #1 On
D 500	Delay .5 sec
O1 0	Output #1 Off
D 500	Delay .5 sec
GOTO 1	Loop to 1$
A$ I1 ON	Alarm on Input #1 On
O2 1	Output #2 Off
B I	Break any active active INPUT cmnd
P “AN ALARM HAPPENED”	Print an Alarm message
D 1000	Delay 1 sec
O2 0	Output #2 Off
END	End Alarm task
VARIABLE$	Variable Input label triggered by ^V
Input “?” X1	Input to X1
END	End Variable Input task

- Fixed LSTERR to work even if MSG=0. Note: LSTERR will return the last error the BDS5 encountered since an Enable (EN) command.
- Fixed another factory initialization problem that would cause a software watchdog trip on entry to the User Program Editor due to the user program being uninitialized in a new BATRAM chip.
- Enhanced the Error Recovery Task to cancel the user INPUT command if it is active when a system error occurs. This prevents the Error Handler from holding further program execution after an error due to the INPUT command having control of the serial port. Normally, the Error Handler first needs to print an error message to the serial port and then pass execution to the ERROR$ user program label if it exists.
- Enhanced the Error Recovery Task to cancel a Task Idling command such as “H” and thus allow the ERROR$ task to properly execute if it exists.

BDS5 FIRMWARE UPGRADE NOTICE

DOC=B5FU305.DOC.R8 / FIRM-UP.DOC JANUARY 27, 1995

VERSION 3.0.5

UPGRADES

When upgrading older systems (pre version 3.0.0) with 3.0.5. Be sure to initialize the three new non-volatile flags for proper operation:

1. **ECHO=1**
2. **MSG=1**
3. **EXTDX=0**

CHANGES (from 3.0.4)

- Added MONITOR=Off support from the user program
  The user can now turn on and off the MONITOR mode under user program control.
- Fixed a problem with EXTLOOP.
  A problem was found where the position error was being calculated when the drive was disabled. This could cause a PE OVERFLOW error if |PEXT-PCMD|>PEMAX. The position error is now only calculated if the drive is enabled.
- Added PCMD=PEXT when EXTLOOP=1 and drive disabled.

  External Position loop mode was enhanced where when the drive is disabled, the variable PCMD is set equal to PEXT. This prevents a possible PE OVERFLOW error when enabling the drive without first executing a NORM command (The NORM command sets PCMD=PEXT when EXTLOOP=1 and it sets PCMD=PFB when EXTLOOP=0).
- Fixed front panel LED initialization on power-up.

  The FAULT LED and ACTIVE LED were being briefly set to the wrong state on power-up. This was corrected.
- Fixed -- A Syntax error in ERROR$ would cause re-execution of ERROR$ (infinite loop).
  Now the BDS5 will break the user program on any syntax error in ERROR$.
- Fixed -- An invalid command (nonexistent) would execute ERROR$ if it existed.
  Now the BDS5 will not allow any program execution until the invalid command has been removed.
- Fixed -- An error occurring while in Monitor mode will print the error message and then return to Monitor mode. Any printing from the ERROR$ routine will be suppressed. Syntax error in ERROR$ would cause re-execution of ERROR$ (infinite loop).

BDS5 TIMING TESTS

INSTRUCTION	Time (msec) Ver 2.0.5	Time (msec) Ver 3.0.5
O1 ON	1.90	1.90
O1 OFF	1.90	1.89
OUT=OUT!0C8h	2.86	2.84
? 1 EQ 1 O1 ON	3.99	3.97
X1=X2	1.72	1.99
X1=X2+1	2.18	2.45
X1=X2-1	2.17	2.45
X1=X2*100	2.33	
X1=X2/100	2.36	2.63
ZPE	1.20	1.69
NORM 0	2.00	2.69
O1 ON (GEAR ON)	2.07	2.03
X1=X2+1 (GEAR ON)	2.37	2.62

* (VDEFAULT 1000) **(Profile calculation time only)
Note : The drive is Enabled (EN) for all timing tests
ACC=DEC=100,000 & SCRV=2

INSTRUCTION	Time ** (msec) Ver 2.0.5	Time ** (msec) Ver 3.0.5
MI 40960 1000	6.00	6.15
MI 40960 1000 (GEAR ON)	6.80	7.02
MI 40960 1000 (GEAR ON & In Motion)	7.00	8.70
MI 40960 1000 (REG ON)	6.20	6.41
MI 40960 1000	7.00	8.26
MI 40960 *	5.00	5.24
MI 40960 * (GEAR ON)	5.80	5.97
MI 40960 * (REG ON)	5.10	5.39

* (VDEFAULT 1000) **(Profile calculation time only)
Note : The drive is Enabled (EN) for all timing tests
ACC=DEC=100,000 & SCRV=2

INSTRUCTION	Time (msec) Ver 2.0.5	Time (msec) Ver 3.0.5
GOSUB 10 &10$	1.60	1.37
GOTO 10 & 10$	1.60	1.58
JT 40960 1000 **	5.80	5.77
JF 40960 1000 **	5.80	5.77
MA 40960 1000 **	5.50	5.16
MA 40960 * **	5.00	4.16
MCI 1000 1000 200 ** MCI 1000 0 MCGO	11.00	11.12
MRD 1000 100 CW **	3.50	3.17
NORM 0	2.00	2.50
PX1	3.50	4.52
PX1[8]	3.50	4.52
P"X1="XI	3.80	5.02
GOSUB 120 & 120$ & RET	3.40	2.56
TIL 1 EQ 0	2.60	2.44
ZPE	1.00	1.55
? 1 EQ 1 O1 ON	4.00	3.92

IF 1 EQ 0
X1=1
ELIF 1 EQ 0
X1=2
ELSE
X1=3
ENDIF
	9.00	9.46

20$	1.00	1.06
J 1000 **	n/a	3.10
MI 40960 1000 **	n/a	6.23
J 1000 (In Motion) **	n/a	3.66

* (VDEFAULT 1000) **(Profile calculation time only)
Note : The drive is Enabled (EN) for all timing tests

TIMING METHOD #1	TIMING METHOD #2
Divide the number of loops counted (X1) into	With an oscilloscope, measure the amount of

10 seconds (10 / X1) to get the single loop time. Perform this operation twice, once with the <instruction> and once without the <instruction>. Subtract the amount of time it takes to run this program with the <instruction> from the time to run without the <instruction>. The result is the amount of time it takes to execute the instruction.	time that Output #1 is ON while running the program with the <instruction> and subtract from this amount of time the Output #1 is ON while running the program without the <instruction>. The result is the amount of time it takes to execute the instruction.
4$ X1=0 TMR1=10000 5$ X1=X1+1 <instruction> ? TMR1 GT 0 GOTO 5 B	5$ PLIM 0 EN O1 ON <instruction> O1 OFF W0 D 1000 GOTO 5

BDS5 SERIAL COMMUNICATIONS 3

DOC=BDS5COM3.DOC Rev 5 JANUARY 31, 1995

ENHANCED COMMUNICATIONS PROTOCOL
(Firmware version 3.1.0)

There is a need for a more robust ASCII communications protocol for tight computer based communications. Beginning with BDS5 firmware version 3.1.0, an enhanced serial communications format will be available. The new protocol adds a serial 8-bit checksum to the end of each BDS5 command. The BDS5 will process this checksum by comparing it to the actual checksum of the command string. If the two checksums match, then the BDS5 will echo an <ack>(06h) character to the serial port and then process the command. If the two checksums do not match, then the BDS5 will echo an <nak> (15h) to the serial port and the command will not be processed. The Serial Checksum mode is enabled by turning on the user flag SCKSUM. The following format will be enabled when the Serial Checksum mode is enabled (SCKSUM=1) :

<command string><checksum><cr>

where :

<command string> is a valid BDS5 ASCII command
<checksum> is an eight bit checksum of the <command string> represented by two ASCII hex characters.

example :

P PFB48<cr> ;print the feedback position
P PFB48<cr><lf><ack>--> ;string received, echoed, acknowledged, and
; prompt returned

Command	"P"	" "	"P"	"F"	"B"	Checksum	
Hex	50h	20h	50h	46h	42h		148h
Decimal	80	32	80	70	66	328	148h

To calculate the checksum, each ASCII byte in the command should be summed excluding the <cr>, then drop all but the least significant byte, this is the serial checksum. So the checksum of "P PFB" is 48.

The only difference between this convention and that of the normal BDS5 is the addition of a checksum. The checksum will consist of a hex byte represented by two ASCII characters (0-9, A-F, or a-f).

If the command is valid and checksums, the BDS5 will execute the command and will respond with :

"<command string><checksum><cr><lf><ack>-->"

where : <ack>=^F=06h=06d

If a checksum error is detected, then the BDS5 will ignore the command and will respond with :

"<command string><checksum><cr><lf><nak>-->"

where : <nak>=^U=15h=21d

A 16-bit checksum of the contents of the BDS5 program memory can also be printed to the serial port. This checksum is contained in the variable :

"PGMCKSUM"

It can be displayed by entering the following command :

"P PGMCKSUM"	;this prints in decimal
"P PGMCKSUMD7"	;this prints in decimal with SCKSUM=1
"P PGMCKSUM[H]"	;this prints inhex
"P PGMCKSUM[H]D7"	;this prints in hex with SCKSUM=1

Notice that"[H]" has a checksum of 256 decimal or 0h, so the checksum of "P PGMCKSUM" is the same as the checksum for "P PGMCKSUM[H]".

Remember that once the Serial Checksum is turned on, it can only be turned off with a valid checksum!. The following string will turn off the Serial Checksum :

"SCKSUM=043<cr>"

where : 43 equals the checksum of the string : "SCKSUM=0"

The Serial Checksum is case sensitive :

P PFB48<cr>	;print the feedback position
p pfbC8<cr>	;print the feedback position

When entering a command with Serial Checksum enabled, backspaces (<bs>) will correct a typo and will not themselves effect the checksum.

Also, after entering the BDS5 internal editor mode (ED), the serial checksum will be momentarily turned off. Upon exiting the editor mode (<Esc>), the BDS5 will resume serial checksum operation. To enter the editor, type "ED89"<cr>.

NEW BDS5 CHECKSUM COMMANDS **(Firmware version 3.1.0 and later)**	
SCKSUM 0|1	Used to enable the serial checksum option. This option will require all serial strings transmitted to the BDS5 to be followed by an eight bit checksum. The format of the checksum is two ASCII hex characters. To turn the Serial Checksum off, the following string must be entered : SCKSUM=043 SCKSUM is set to 0 on Power-up.
PGMCKSUM	This is a 16-bit checksum of the BDS5 User Program.

RS-485 Communications with Serial Checksum

RS-485 can be used in conjunction with the new serial checksum feature for enhanced communications.

Example with MultiDrop enabled :

```
SCKSUM=1<cr>                    ;enable Serial Checksum protocol
        SCKSUM=1<cr><lf>        ;echo
        -->                     ;prompt

PLIM=09F<cr>                    ;turn off software travel limits
        PLIM=09F<cr><lf>        ;echo
        <ack>-->                ;acknowledge and echo

EN93<cr>                        ;enable the motor
        EN93<cr><lf>            ;echo
        <ack>-->                ;acknowledge and echo

ADDR=65C3                       ;enable RS-485 and enable address"A"
                                ;       (ASCII 65 decimal)
        ADDR=65C3<cr><lf>       ;echo
        <ack>A->                ;Axis ID set, axis then disconnects until addressed

\A<cr>                          ;select axis "A"
        <cr><lf>                ;echo
        <ack>A->                ;acknowledge and echo

MI 4096 1000<cr>                ;move incremental 4096 counts @ 1000 rpm
                                ;       - no checksum
        MI 4096 1000<cr><lf>    ;echo
        <nak>A->                ;negative acknowledge - bad checksum

MI 4096 10006A<cr>              ;move incremental 4096 counts @ 1000 rpm with
                                ;       - checksum of 6A
        MI 4096 10006A<cr><lf>      ;echo
        <ack>A->                ;acknowledge

SCKSUM=043<cr>                  ;turn off serial checksum
        SCKSUM=043<cr><lf>      ;echo
        <ack>A->                ;acknowledge and echo

ADDR=0<cr>                      ;turn off RS-485
        <cr><lf>                ;back to standard configuration

PROMPT=1<cr>                    ;turn on prompts
        -->                     ;RS-232 prompt

ECHO=1<cr>                      ;turn on echo
        -->                     ;prompt

P PFB<cr>                       ;print feedback position
        P PFB<cr><lf>           ;echo
                1212            ;12 position number (leading spaces)
        -->                     ;prompt
```

Additionally, maximum through put of data can be achieved by turning off the user flags ECHO and PROMPT. These two flags will suppress serial port character echoing and the BDS5 prompt string.

The Echo flag when set to 0 will suppress echoing of all characters received by the BDS5 (this includes the <cr><lf>termination sequence).

Prompt flag when set to 0 will suppress the BDS5 three character prompt when it is set to 0.

Please note that the BDS5 uses the prompt (-->, ==> etc.) to indicate to the user that it is ready for another command. This means that until the prompt is transmitted, BDS5 is not listening to the serial port. When the BDS5 prompt is suppressed, it may be necessary to wait a few milli-seconds after each command before transmitting the next command.

With both **ECHO=0** and **PROMPT=0**, the reply string for :

<command string><checksum><cr>

would be : **<ack>** or **<nak>**

Example with MultiDrop enabled, PROMPT=0 and ECHO=0 :

```
PROMPT=0<cr>                    ;turn off prompts
      PROMPT=0<cr><lf>          ;no prompt returned

ECHO=0<cr>                      ;turn off character echo
      ECHO=0<cr><lf>            ;

SCKSUM=1<cr>                    ;enable Serial Checksum protocol
                                ;nothing echoed or returned!

PLIM=09F<cr>                    ;turn off software travel limits - checksum = 9F
      <ack>                   ;command acknowledged

EN93<cr>                        ;enable the motor - checksum = 93
      <ack>                   ;command acknowledged

ADDR=65C3                       ;enable RS-485 and enable address "A" (ASCII 65
                                ;       decimal) - checksum = C3
      <ack>                   ;command acknowledged - Axis ID set, axis then
                                ;       disconnects until addressed

\A<cr>                          ;select axis "A" (checksum not required for axis select)
      <ack>                   ;command acknowledged

MI 4096 1000<cr>                ;move incremental 4096 counts @ 1000 rpm with
                                ;       no checksum
      <nak>                   ;command not-acknowledged

MI 4096 100055<cr>              ;move incremental 4096 counts @ 1000 rpm with
                                ;       checksum of 55 (bad)
      <nak>                   ;command not-acknowledged



MI 4096 10006A<cr>              ;move incremental 4096 counts @ 1000 rpm with
                                ;       checksum of 6A
      <ack>                   ;command acknowledged

ADDR=0<cr>                      ;turn off RS-485
```

```
PROMPT=1<cr>                    ;turn on prompts
        -->                     ;RS-232 prompt

ECHO=1<cr>                      ;turn on echo
        --.                     ;prompt

P PFB<cr>                       ;print feedback position
        P PFB<cr><lf>           ;echo
                1212            ;12 position number (leading spaces)
        -->                     ;prompt
```

PRINT KEY ---- The following filenames have been assigned :

PAGE 1 OF 2 PAGES

DESCRIPTION	**FILENAME**
Cover / Title Page	AFM1 .DOC
Technical Manual Configuration / Configuration Table	AFM2 .DOC
Customer Response	AFM3 .DOC
Copyright Page	AFM4 .DOC
Foreword	AFM5 .DOC
How to Use This Manual	AFM6 .DOC
Table of Contents	AFM7 .DOC
List of Figures	AFM8 .DOC
List of Tables	AFM9A .DOC
Chapter 1	CH1 .DOC
Chapter 2	CH2 .DOC
Chapter 3	CH3 .DOC
Chapter 4	CH4 .DOC
Chapter 5	CH5 .DOC
Chapter 6	CH6 .DOC
Appendix A	FAPPA .DOC
Appendix B	FAPPB .DOC
Appendix C	FAPPC .DOC
Appendix D	FAPPD .DOC
Appendix E	FAPPE .DOC
Appendix F	FAPPF .DOC

PRINT KEY ---- The following filenames have been assigned :

PAGE 2 OF 2 PAGES

DESCRIPTION	**FILENAME**
Glossary	GLOSS .DOC
Index	INDEX .DOC
BDS5 Upgrade Notices	NUGCVR .DOC
	NUGHIST .DOC
	NUPGRD1 .DOC
	NUPGRD2 .DOC
	NUPGRD3 .DOC
	NUPGRD4 .DOC
	NUPGRD5 .DOC

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP